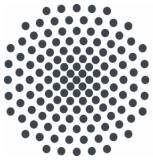

Universität Stuttgart

Integration von mechanischen Arbeitsprozessen in den modellbasierten Systementwurf

Abschlussarbeit

zur Erlangung des akademischen Grades
Master of Science (M.Sc.)

an der

Universität Stuttgart
Institut für Statik und Dynamik
Studiengang Luft-und Raumfahrttechnik

Verantwortlicher Hochschullehrer und Betreuer: PD. Dr.-Ing. habil. Stephan Rudolph
Externer Betreuer: Philipp M. Fischer

Eingereicht von: Katharina Roventa
Matrikelnummer: 3203435
Datum der Abgabe: 28.03.2018

Vorwort und Danksagung

Die vorliegende Abschlussarbeit entstand während meiner Tätigkeit beim Deutschen Zentrum für Luft- und Raumfahrt (DLR) in Braunschweig in der Abteilung für Simulations - und Softwaretechnik.

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Besonders gebührt mein Dank meinem betreuenden Hochschullehrer Herr Stephan Rudolph und meinem externen Betreuer Herr Philipp Martin Fischer, die meine Masterarbeit betreut und begutachtet haben. Für die hilfreichen Anregungen und die konstruktive Kritik bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken.

Braunschweig, den 28.03.2018

Eidesstaatliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Braunschweig, den 28.03.2018

Katharina Roventa

Abstract

Die vorliegende Arbeit befasst sich mit der Erstellung, Implementierung und Untersuchung einer bidirektionalen Schnittstelle zwischen der DLR Software Virtual Satellite und der Computer-Aided-Design Software CATIA. Im Vordergrund steht der Austausch zwischen modellbasiertem Systementwurf und dem mechanischen Design in Bezug auf Raumfahrtmissionen. Durch die Schnittstelle soll das Verständnis der mechanischen Konfiguration der Raumfahrtmission, mit Hilfe einer engen Anbindung an das Systems Engineering verbessert werden. Dabei soll der mechanische Entwurfsprozess an den übergeordneten modellbasierten Systementwurfsprozess angekoppelt werden und entlang des gesamten Lebenszyklus, eine Analyse der Entwurfsprozesse und deren Kopplungen stattfinden. Aus dieser Analyse lassen sich Anforderungen an den Detailierungsgrad und den tatsächlichen Informationsgehalt der System Modelle und der mechanischen Modelle ableiten. Ferner wird diese Information benötigt, um ein geeignetes Mapping auf konzeptioneller Ebene und Systemebene zu definieren, welches einen Informationsaustausch ermöglicht.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AT	Assembly Tree
CA	Category Assignment
CAD	Computer-Aided-Design
CE	Concurrent Engineering
CEF	Concurrent Engineering Facility
CDM	Conceptual Data Model
CT	Configuration Tree
DLR	Deutsches Zentrum für Luft- und Raumfahrt
EC	Element Configuration
ECSS	European Cooperation for Space Standardization
ED	Element Definition
EGS-CC	European Ground Systems Common Core
EO	Element Occurence
ESA	Eurpean Space Agency
ER	Element Realization
GSEL	Generic System Engineering Language
json	Javascript Object Notation
MBSE	Modelbased Systems Engineering
PT	Product Tree
RCP	Rich Client Platform
RPC	Remote Procedure Call
SE	Systems Engineering
ST	Storage Tree
STEP	STandard for the Exchange of Product model data
STL	Standard Tessellation Language
SysML	Systems Modeling Language
VCS	Version Control System
VSD	Virtual Spacecraft Design
UUID	Universally Unique Identifier
XML	Extensible Markup Language

Terminologie

CATIA

CATIA (Computer Aided Three-Dimensional Interactive Application) ist ein CAD-System, das von der französischen Firma Dassault Systèmes entwickelt wurde. Ursprünglich für den Flugzeugbau entwickelt, hat sich die Software heute in verschiedenen Branchen zur Analyse und Visualisierung des mechanischen Designs etabliert. [48]

Modell

Ein Modell stellt eine vereinfachte Version eines Konzepts, Systems oder einer Struktur dar. Das Modell kann dabei eine grafische, mathematische oder physikalische Repräsentation des Systems darstellen. Bei einem Modell handelt es sich immer um eine Abstraktion der Realität, das bedeutet, dass nur relevante Informationen im Modell vorhanden sind.[47] Durch ein Modell soll das Verständnis des Gesamtsystems verbessert werden, in dem das System mithilfe des Modells erklärt wird. Zudem soll es bei der Fragestellung von ”Was wäre wenn” - Szenarien helfen und Systementscheidungen unterstützen.[20]

System Modell

Das System Modell ist eine strukturierte Repräsentation des Systems, das sich auf allumfassende Systemspezifikationen, sowie Verhalten, Eigenschaften, Interaktionen und die Struktur des Systems, fokussiert.[20]

VCS

Eine Versionsverwaltung oder auch *Version Control System* (VCS) genannt, ist ein System, das zur Erfassung von Änderungen an Dokumenten oder Dateien verwendet wird. Die Hauptaufgaben bestehen im Wesentlichen darin, Änderungen zu protokollieren, eine Wiederherstellung von älteren Daten zu garantieren und die Archivierung verschiedener Version vorzunehmen.

UUID

Ein *Universally Unique Identifier* (UUID) wird in der Softwareentwicklung verwendet und stellt einen Standard für Identifikatoren dar. Mit Hilfe einer UUID ist es möglich Informationen in verteilten Systemen ohne zentrale Koordination eindeutig kennzeichnen zu können. [30]

Kapitel 1

Einleitung

Der modellbasierte Systementwurf kommt beim DLR während Concurrent Engineering (CE) Studien zum Einsatz. Hier wird mit Hilfe der internen DLR Software Virtual Satellite in kürzester Zeit ein System Modell des gerade geplanten Raumfahrzeugs erstellt.[13] Während dieser CE Studien hat sich gezeigt, dass es in verschieden Bereichen zu Schwierigkeiten führt, wenn es darum geht, das Wissen der einzelnen Experten im Team zu teilen und zu beschreiben. Dabei ist besonders die mechanische Konfiguration eine Herausforderung, da dort zu Konfigurationszwecken Positionen und Orientierung verschiedener Bauteile nur verbal beschrieben werden. Aus diesem Grund wurde die MBSE Software Virtual Satellite erweitert, damit jeder Ingenieur selber definieren kann wie seine Bauteile aussehen und wo sie im Raumfahrzeug zu platzieren sind. Mit einer entsprechenden Visualisierung konnte das gemeinschaftliche Verständnis für die Konfiguration gesteigert werden.[35]

Dennoch zeigt sich, dass diese Arbeitsweise allein einen Bruch mit den klassischen Arbeitsprozessen im Bereich des mechanischen Designs bedeutet. Da Werkzeuge wie das CAD Tool, CATIA, als gesetzt gelten, müssen diese auch in die einzelnen Arbeitsprozesse im MBSE angekoppelt werden. Die komplette bidirektionale Anbindung an einen raumfahrtbezogenen MBSE Prozess ist bis heute jedoch nicht ausreichend analysiert und umgesetzt.

1.1 Zielsetzung der Arbeit

Ziel der Arbeit ist es die gemeinsamen Anforderungen für die Kopplung des modellbasierten Systementwurfs und des mechanischen Entwurfsprozesses festzuhalten und zu Anforderungen an eine Schnittstelle abzuleiten. Dort stehen insbesondere die Anforderungen an die Bidirektionalität der Schnittstelle im Vordergrund. Die Umsetzung der Schnittstelle erfolgt zwischen der MBSE Datenbank Virtual Satellite und der CAD Software CATIA. Bestehende Ansätze, zur Umsetzung der Kopplung von mechanischem Design an den modellbasierten Systementwurf, wie sie in der Vorgängerversion der Software Virtual Satellite oder Virtual Spacecraft Design, einer Software von der europäischen Raumfahrtbehörde ESA [15], umgesetzt wurden, sind bisher nur unidirektional.

In dieser Arbeit wird aufbauend auf den bereits vorhandenen Konzepten eine bidirektionale Umsetzung erfolgen. Nach Erstellen der Anforderungen werden einzelne bereits bestehende Austauschverfahren der CAD Software betrachtet und verglichen. Unter Berücksichtigung bestimmter Kriterien werden darauffolgend Austauschverfahren ausgewählt bzw. neu entwickelt und in Virtual Satellite implementiert. Dabei

soll ein möglichst einfaches und einheitliches Verfahren gewählt werden, welches zusätzlich den zukünftigen Austausch an weiteren Informationen ermöglicht. Die anschließende Evaluierung und Analyse des Entwurfsprozesses und dessen Kopplungen wird anhand einer, in der MBSE Software Virtual Satellite, selbst erstellten Testmission vorgenommen.

Die Abbildung 1.1 zeigt den Architekturplan der vorliegenden Masterarbeit. Das System Modell, welches von Virtual Satellite bereitgestellt wird, steht im Austausch mit dem Version Control System und den einzelnen Domänen. Der Begriff der Domäne wird in dieser Arbeit für die einzelnen Subsysteme einer Raumfahrtmission verwendet. Ein Satellit bzw. eine Raumfahrzeug lässt sich in mehrere Subsysteme gliedern, unter anderem in das Thermalsystem, das Lageregelungssystem, das Energiesystem, das Kommunikationssystem, das Datenverarbeitungssystem und das Struktursystem.[52] Durch die Anbindung von CATIA an das System Modell soll es möglich sein, Informationen, die CATIA bereitstellt, für mögliche Domänen Modelle abzuleiten und das neu erstellte System Modell in die dafür vorhergesehene Datenbank abzulegen. Dabei ist es wichtig zu wissen, welche Informationen zwischen dem CATIA Modell und dem System Modell ausgetauscht werden müssen bzw. welche Informationen benötigt werden, um diese später in domänenspezifischen Prozessen zu nutzen. Die Informationen der einzelnen Domänen werden vom verantwortlichen Domänen-Team vorher festgelegt bzw. hergeleitet. Ein Domänen Modell kann dabei beispielsweise ein Modell zur Thermalsimulation oder ein Modell zur Strukturanalyse darstellen.

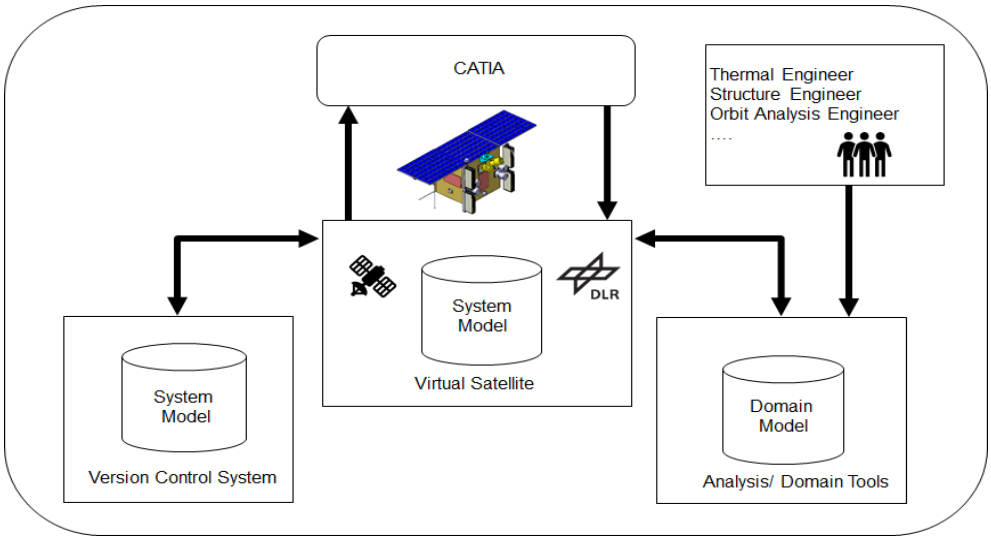

Abbildung 1.1: Architekturplan der Masterarbeit

Um die Relevanz von mechanischen Konstruktionsinformationen aus dem CAD Modell im System Modell zu verdeutlichen, dienen die nachfolgenden kurzen Beispiele. Die genannten Beispiele werden in der Masterarbeit nicht an das System Modell gekoppelt, diese und ähnliche Fälle werden aus bestehender Literatur und „Best Practices“ methodisch analysiert, um die spezifischen Informationen, die ausgetauscht werden müssen, zu bestimmen.

Wichtige Informationen, die ein Thermalingenieur aus dem CAD-Modell ziehen kann sind beispielsweise Position, Ausrichtung und Verbindung der einzelnen Bauteile des Satelliten, welche in das System Modell mit einfließen. Aus dem von CATIA abgeleiteten bzw. neu generierten System Modell soll es möglich sein genügend Informationen herzuleiten um für eine, in diesem Fall, Thermalsimulation die Simulatoren korrekt zu konfigurieren. Der Thermalingenieur kann damit ein Domänen Modell für den Wärmeaustausch erstellen, modifizieren oder auch verifizieren, nämlich das Thermalmodell des Satelliten. Dabei spielen Fragen eine Rolle wie zum Beispiel: Wie sieht der Wärmeaustausch zwischen den Bauteilkomponenten aus? Wie ist der Gesamtwärmehaushalt des Satelliten? Müssen die Bauteile anders angebracht werden um den Wärmehaushalt im Gleichgewicht zu halten? Besonders in Bezug auf eine Thermalanalyse kann eine solche Simulation sehr hilfreich sein, da der Satellit nicht vorher im Weltraum getestet, sondern nur durch Simulation gezeigt werden kann, dass das Thermalmodell des Satelliten den Anforderungen der Raumfahrtmission und der Weltraumumgebung genügt.

Die mechanischen Designinformationen aus dem CAD-Modell, welche im System Modell hinterlegt werden, sind dabei nicht ausschließlich für die Thermalanalyse von großer Bedeutung, sondern spielen auch eine Rolle bei Strahlungsanalysen, Strukturanalysen oder dem virtuellen Zusammenbau (Virtual Assembly) eines Satelliten. Im Fall des Virtual Assembly, stehen nachfolgende Fragen im Vordergrund: Welches Bauteil soll zuerst angebracht werden? Wie sind die einzelnen Bauteile zueinander ausgerichtet? Wie sind die Bauteile verkabelt und wo werden die Verbindungen lang geführt?

Eine Verbindung, wie im obengenannten Beispiel, zwischen Struktur- und Thermalmodell oder auch zwischen Strukturmodell und Virtual Assembly, kann direkt abgeleitet werden. Somit ist es möglich, dass einzelne Domänen an einem System Modell arbeiten und den jeweiligen aktuellen Stand jeder Domäne im System Modell berücksichtigen können. Das verstärkt zum einen das Verständnis der jeweiligen Domänen zueinander und zum anderen ist es möglich ein System Modell zu erstellen, welches aus Sicht aller Domänen auf dem aktuellsten Stand ist und in der Lage ist Simulatoren für nötige Simulationsanalysen zu konfigurieren.

1.2 Aufbau der Arbeit

Während das vorliegende Kapitel die Ausgangsstellung für diese Masterarbeit beinhaltet und die Zielsetzung erläutert, werden in den weiterführenden Kapiteln folgende Themen behandelt:

Kapitel 2: In diesem Kapitel werden das Systems Engineering und das modelbased Systems Engineering im Allgemeinen vorgestellt, bevor im Speziellen auf die Anwendung des modelbased Systems Engineering im Raumfahrzeugentwurf eingegangen wird. Des Weiteren werden wichtige Werkzeuge im Systems Engineering vorgestellt.

Kapitel 3: Dieses Kapitel stellt die MBSE Software Virtual Satellite des Deutschen Zentrums für Luft- und Raumfahrt vor. Dabei werden Aufbau, Nutzung und Entwicklungsumgebung der Software konkretisiert.

Kapitel 4: Dieses Kapitel stellt das mechanische Design und die CAD Software CATIA und dessen Gebrauch im Entwurf einer Raumfahrtmission explizit vor. Des Weiteren wird auf bestehende Methoden der Integration von mechanischen Design in den modellbasierten Systementwurf eingegangen.

Kapitel 5: In diesem Kapitel wird eine Gegenüberstellung zwischen Ziel und Stand der Technik vorgenommen. Des Weiteren werden die Vorteile einer bidirektionalen Integration aufgezeigt und der Informationsfluss der bidirektionalen Schnittstelle diskutiert.

Kapitel 6: Dieses Kapitel befasst sich mit der Umsetzung und Implementierung der bidirektionalen Schnittstelle unter der in Kapitel 5 vorgestellten Kriterien anhand der Software Virtual Satellite und der Software CATIA. Dabei wird sowohl die Implementierung aus CATIA Sicht, als auch aus Virtual Satellite Sicht vorgestellt.

Kapitel 7: Dieses Kapitel analysiert anhand eines Beispieldatensatzes den Informationsfluss der bidirektionalen Schnittstelle zwischen CATIA und Virtual Satellite. Dabei werden verschiedene Anwendungsszenarien dargelegt.

Kapitel 8: Abschließend wird in diesem Kapitel eine Zusammenfassung der Arbeit und der erlangten Ergebnisse durch den Anwendungsfall gegeben.

Kapitel 9: Aufbauend auf den Ergebnissen aus Kapitel 7 wird in diesem Kapitel ein Fazit gezogen und ein Ausblick auf die weitere Nutzung der bidirektionalen Schnittstelle gegeben.

Kapitel 2

Einführung in den modellbasierten Systementwurf

Modellbasierter Systementwurf, kurz *MBSE*, basiert auf der Idee den Systementwurf über ein digitales Modell des Systems zu unterstützen. Alle Informationen die in dem System für mehrere Domänen von Relevanz sind, werden in diesem Datenmodell abgespeichert und miteinander in Bezug gebracht. Das Modell dient dabei als eine *Single Source of Truth*. Alle Domänen teilen und beziehen ihr Wissen über dieses eine digitale Modell.[20] Auf den folgenden Seiten wird eine Einführung in den Systementwurf und den modellbasierten Systementwurf gegeben. Dabei werden sowohl Werkzeuge des Systementwurfs vorgestellt, als auch die Nutzung von Systementwurf und MBSE in der Raumfahrt dargestellt.

2.1 Systementwurf und dazugehörige Werkzeuge

Der Systementwurf, oft auch als Systems Engineering (SE) bezeichnet, ist ein interdisziplinärer Ansatz mit dem Ziel, erfolgreiche Systeme zu realisieren. Das Systems Engineering konzentriert sich auf die Definition und Dokumentation der Systemspezifikationen in der frühen Entwicklungsphase, sowie die Erarbeitung eines Systemdesigns und die Überprüfung des Systems auf Einhaltung der gestellten Anforderungen unter Berücksichtigung des Gesamtproblems: Betrieb, Zeit, Test, Erstellung, Kosten, Planung, Support und Entsorgung. [21] [32] Das Systems Engineering integriert alle Disziplinen und bildet einen strukturierten Entwicklungsprozess vom Konzept über die Produktion- bis hin zur Betriebsphase. Es werden sowohl die technischen als auch die wirtschaftlichen Aspekte betrachtet, um ein System zu entwickeln, dass den Benutzerbedürfnissen entspricht.

Systems Engineering ist ein durchgängiger disziplinübergreifender Ansatz zur Entwicklung interdisziplinärer Systeme, wie zum Beispiel der Entwicklung eines Raumfahrzeuges. Die zunehmende Vernetzung und Einbettung des Systems erfordert eine ganzheitliche Herangehensweise, um zwischen allen beteiligten Disziplinen (Software, Hardware, usw.) ein gemeinsames Systemverständnis zu erlangen.[32]

2.1.1 V-Modell

Bei der Betrachtung des Systems Engineering spielt das sogenannte V-Modell eine wichtige Rolle. Das V-Modell ist eine grafische Darstellung des Systementwurfszyklus. Es zeigt die klassischen Schritten, welche

ein System im Entwurfszyklus durchläuft. Die klassischen Schritte beinhalten dabei das technische Planen des Systems, den Aufbau der Systemarchitektur, die Anforderungserfassung, aber auch das detaillierte Design, sowie die Entwicklung und Verifikation des Systems. [5] Besonders die ersten beiden Elemente spielen eine zentrale Rolle im Systementwurfszyklus, da diese alle nachgehenden Phasen mitbestimmen. [29] [25] In Abbildung 2.1 werden die Schritte des Systems Engineering im V-Modell veranschaulicht. Zu sehen ist auch die Zeitskala für die Erstellung des Systems, welche direkt aus dem V-Modell entnehmbar ist.

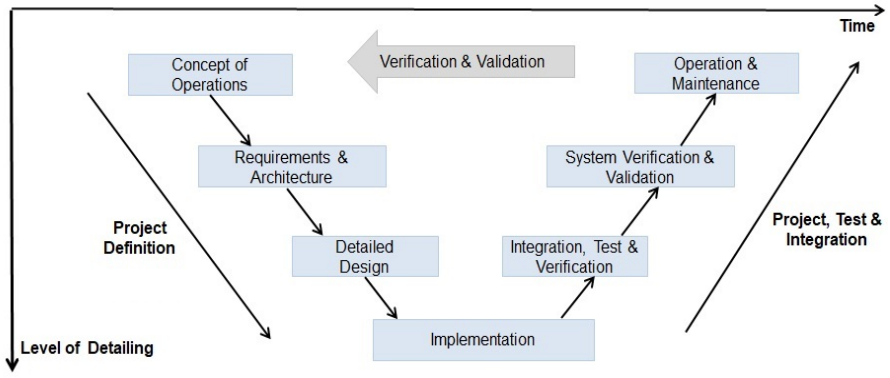

Abbildung 2.1: V-Modell

Dabei fasst das V-Modell die wichtigsten Schritte zusammen, die in Verbindung mit den entsprechenden Ergebnissen innerhalb der Projektentwicklung zu treffen sind. Es beschreibt die durchzuführenden Aktivitäten und die Ergebnisse, die während der Produktentwicklung erzielt werden müssen. Die linke Seite des "V" repräsentiert die Zerlegung von Anforderungen und die Erstellung von Systemspezifikationen. Die rechte Seite des "V" stellt die Integration, Verifizierung und Validierung dar. Die Validierung kann dabei durch die Frage "Bauen wir das Richtige?" und die Verifizierung durch die Frage "Bauen wir es richtig?" ausgedrückt werden? [5]

In Bezug auf den modellbasierten Systementwurf ist eine Entwicklungsmethodik auf Grundlage des V-Modells geeignet, da das V-Modell auf bewährten konventionellen Entwicklungsmethoden aufbaut und ergänzend durch Methoden und Techniken des modellbasierten Systementwurfs zur Systemanalyse, Systementwicklung und Systemintegration unterstützt werden kann.[1]

2.1.2 Concurrent Engineering (CE)

Ein weiteres wichtiges Werkzeug im Systems Engineering ist neben dem MBSE auch das Concurrent Engineering, das in Bremen beim DLR in der Concurrent Engineering Facilitiy (CEF) praktiziert wird. [13] Concurrent Engineering Facilities sind spezialisierte Einrichtungen mit dem Ziel Systeme erfolgreich durch schnelle Design Iterationen zu erstellen. Das wird durch die Zusammenstellung eines Teams von Experten

(entweder physisch oder virtuell) in einem Raum mit einem fokussierten Designziel erreicht. Die Ingenieure bekommen dabei eine feste Zeitvorgabe von wenigen Wochen um das Designziel zu erreichen.[51] Mittels Concurrent Engineering ist es dem Team möglich in einem iterativen Prozess synchron und koordiniert miteinander zu arbeiten bis ein einheitliches Übereinstimmen und erfolgreiches Konzept für das System erstellt wurde. Daraus entsteht ein System Modell, das konsistent und für alle zugänglich ist, und während des gesamten Projektlebenszyklus verwendet werden kann. [24]

2.1.3 Systems Modeling Language (SysML)

Im Bereich des Systems Engineering findet auch die*Systems Modeling Language* (SysML) ihre Anwendung. Dabei handelt es sich um eine grafische auf UML 2 (Unified Modelling Language) basierende, standardisierte Modellierungssprache. Diese wurde von der *Object Managment Group* (OMG) in Zusammenarbeit mit dem *International Council on Systems Engineering* (INCOSE) entwickelt. Sie wird im Systementwurf zur Modellierung von komplexen Systemen genutzt [22] und unterstützt dabei die Analyse, das Design und den Test von komplexen Systemen. SysML wird verwendet um alle Aspekte eines Systems entweder direkt oder über eine Schnittstelle mit einem anderen Modell zu modellieren. Dabei werden die SysML Diagramme dazu genutzt um Anforderungen, Strukturen, Verhaltensweisen und Parameter vom System bis zur Komponentenebene zu beschreiben. Mittels verschiedener SysML Diagrammtypen können Systemanforderungen modelliert und Systeminformationen zwischen unterschiedlichen Stakeholdern ohne Verständnisprobleme ausgetauscht werden.[20]

2.2 Modellbasierter Systementwurf (MBSE)

Modellbasierter Systementwurf, im englischen modelbased Systems Engineering (MBSE), baut auf dem Systems Engineering auf und basiert auf dem Prinzip eines einheitlichen System Modells das disziplinspezifische Modelle über den gesamten System Lebenszyklus hinweg koordiniert. Es handelt sich um ein modernes Paradigma des Systems Engineering, das genutzt wird um SE Prozesse zu verbessern, indem das System als ein Satz von integrierten Computermodellen beschrieben wird. Dieses Paradigma kann dabei helfen von der dokumentenzentrierten Vorgehensweise, wie es im klassischen Systems Engineering der Fall ist zu einer modellbasierten Entwicklungsmethodik zu gelangen.[20]

Das Ziel von MBSE ist das Entwerfen und Managen eines einzigen einheitlichen System Modells, welches alle für das System relevanten Aspekte enthält. Zur Beherrschung der Komplexität in der Entwicklung intelligenter technischer Systeme ist eine durchgängige konsistente Virtualisierung der Produktentstehung unabdingbar.[32] Zur Umsetzung dieser Durchgängigkeit werden zwei Ansätze verfolgt: das direkte Koppeln der Modelle oder der Aufbau eines disziplinübergreifenden System Modells. Die Vorteile dieser

Methode liegen in der transparenten Verfolgung des Lebenszyklus eines Systems, der Konsistenz, die zwischen den verschieden Disziplinen liegt und der Automatisierungsmöglichkeit des Informationstransfers zu den einzelnen domänenspezifischen Tools.[39]

2.2.1 Modellbasierter Systementwurf in der Raumfahrt

Das modellbasierte Systems Engineering spielt eine besonders wichtige Rolle bei sehr komplexen und interdisziplinären Systemen, wie sie in der Raumfahrtbranche bestehen. Der Entwurf eines Satelliten oder eines Raumfahrzeugs involviert eine Vielzahl an Ingenieursdisziplinen, wie z.B. die mechanische, die thermische oder die elektrische Disziplin. Um die Komplexität in einem überschaubaren Rahmen zu halten bietet sich ein einheitliches System Modell an, von welchem aus alle beteiligten Ingenieurdomänen Zugriff haben um eigene domänenspezifische Informationen einzubauen.

Betrachten wir ein Raumfahrzeug als solches System wird auch dieses in verschiedene Ingenieurdomänen aufgeteilt. Jede Domäne beinhaltet dabei, die für sie relevanten Informationen und wird in der Raumfahrt auch als Subsystem bezeichnet. Subsysteme können beispielsweise das Thermalsystem, das Lageregelungssystem oder auch das Kommunikationssystem darstellen. [52] Diese Subsysteme sind stark voneinander abhängig und jede signifikante Änderung in einem Subsystem kann gleichzeitig auch eine Änderung in einem anderen Subsystem bedeuten. Beispielsweise hat das Subsystem der Lageregelung und dessen Komponenten, wie Aktuatoren, Einfluss auf das Subsystem der Energieversorgung, da die Aktuatoren in den meisten Fällen mit Energie versorgt werden müssen. Um die Informationen der Domänen im System konsistent und einheitlich zu halten, sowie für jeden zugänglich zu machen kann MBSE genutzt werden.

Das Entwerfen, Konstruieren und Betreiben eines Raumfahrtzeuges wird in verschiedene Phasen unterteilt. Welche unter dem Begriff Lebenszyklusphasen fallen und unter anderem durch Standards der *European Cooperations for Space Standardization (ECSS)* beschrieben werden können. [14] Abbildung 2.2 verdeutlicht die Komplexität eines Raumfahrzeugentwurfs. Während das System Modell im Laufe der einzelnen Phasen immer größer bzw. detailreicher wird, so steigt auch der Informationsaustausch zwischen den einzelnen Ingenieurdomänen.

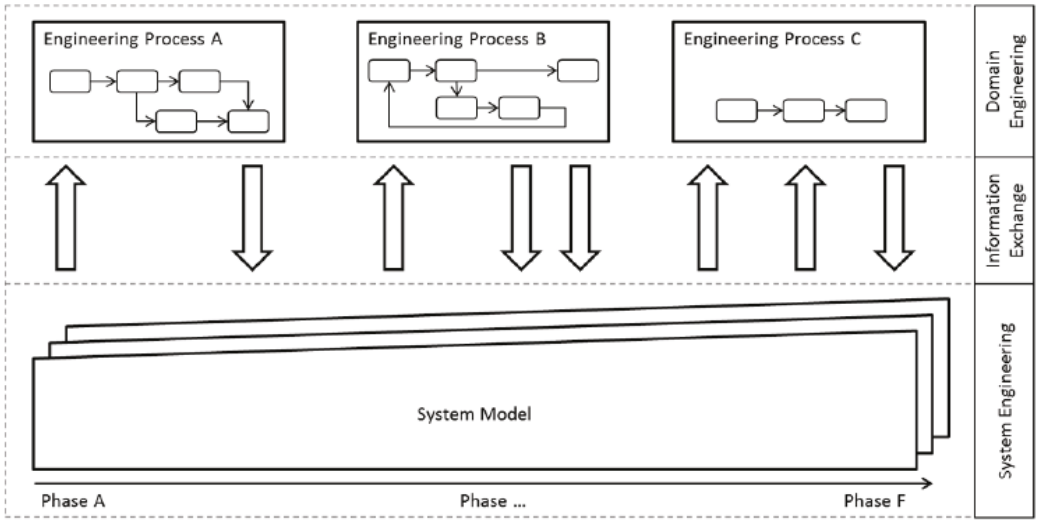

Abbildung 2.2: Einsatz von MBSE in der Raumfahrt[14]

In der folgenden Abbildung 2.3 sind die einzelnen Phasen und deren Zusammenhang mit dem modellbasierten Systementwurf noch einmal konkreter dargestellt. Nach der Vorbereitungsphase, Phase 0, und den ersten Überlegungen zur Umsetzung der Mission, geht es weiter mit Phase A, der Machbarkeitsphase. Dort kommt auch der modellbasierter Systementwurf zum Einsatz und wird fortlaufend bis Phase E genutzt. In der letzten Phase, der Entsorgungsphase F, spielt der modellbasierte Systementwurf keine Rolle mehr, da dort kein *Entwerfen* in dem Sinne stattfindet, sondern die Mission an ihrem Ende angelangt ist. Während zwischen Phase A und D das Design und die Entwicklung des Raumfahrzeuges im Vordergrund stehen, befindet sich das Raumfahrzeug in der Operations- bzw. Betriebsphase E bereits im Missionseinsatz. Jede Phase beinhaltet dabei bestimmte Anforderungen, die es zu erfüllen gilt. Phase A beschäftigt sich mit der Machbarkeitsstudie der Mission, während Phase B sich mit dem detaillierten Entwurf beschäftigt. Phase C und D beinhalten die Entwicklung, sowie den Zusammenbau, die Integration und verschiedene raumfahrtspezifische Tests.[14]

Der modellbasierte Systementwurf in den einzelnen Phasen dient zur Einführung eines System Modells, das Designinformationen konsistent hält und zugänglich macht. Designinformationen einer Raumfahrtmission werden über den gesamten Lebenszyklus hinweg gesammelt und unter den Domänen ausgetauscht. Durch die Verwendung von MBSE können die einzelnen Domänen nun mit einander kommunizieren, effektiver miteinander arbeiten und Komplikationen zwischen den Subsystemen früher erkennen und eliminieren, da alle Domänen mit einem einheitlichen System Modell arbeiten. Normalerweise wird ein System im klassischen Systementwurf dokumentenbasiert dargestellt, mittels MBSE entsteht eine visuelle Darstellung, diese erleichtert das Verständnis des System Gesamtkonzepts.[32] Die Software Virtual Satellite, welche vom Deutschen Zentrum für Luft- und Raumfahrt entwickelt worden ist, stellt eine sogenannte modellbasierte Datenbank dar, die es ermöglichen soll, ein ganzheitliches konsistentes System Modell

für alle Domänen zugänglich zu machen. [13] Auf die Software und deren Nutzung von MBSE wird im nachfolgenden Kapitel 3 näher eingegangen.

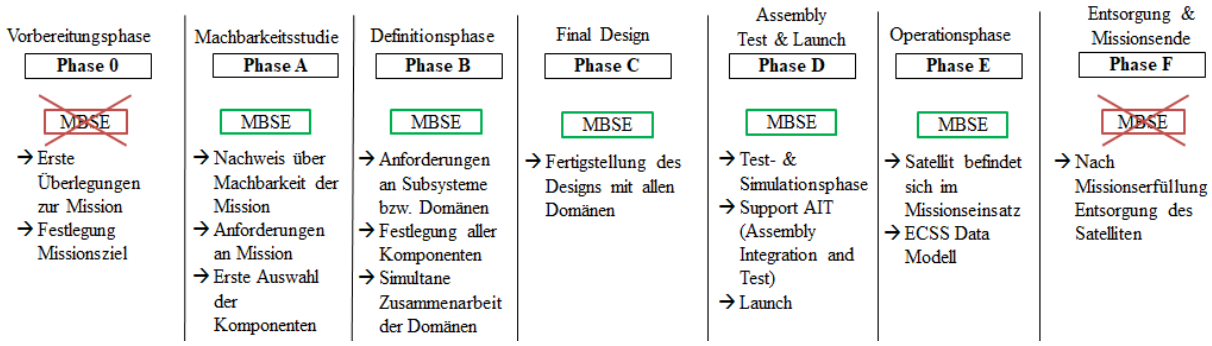

Abbildung 2.3: Einsatz von MBSE in den Lebenszyklusphasen einer Raumfahrtmission

Durch die Nutzung von modellbasierten Systementwurf über den gesamten Lebenszyklus können auch Integrationstests, wie sie bei der Kommunikation in Phase D, der *AIT*(Assembly, Integration and Test) Phase stattfinden, unterstützt werden. Während in der Integration schon gezeigt wurde, dass das Telemetrie System getestet und somit funktioniert, könnte in der Betriebsphase nach ECSS Standards, das Kommunikationssystem integriert werden ohne nochmal komplett neu getestet zu werden, da die Verifizierung schon durch den MBSE erfolgte.[14] Dieses Ziel wird unter anderem vom *European Ground Systems Common Core* (EGS-CC) verfolgt. [37]

MBSE kann ebenfalls die Konfiguration von Simulatoren unterstützen. Dort liefert das detaillierte System Modell entscheidende Informationen für die Simulatorkonfiguration wie eventuelle Verhaltensbeschreibungen. Durch geschickte Transformationen können Simulatorkonfigurationen bei Änderungen des System Modells angepasst werden. Das führt dazu, dass Fehler, die aus manuellen Änderungen bzw. Anpassungen entstehen können, umgangen werden.[44]

2.2.2 Datenbanken im modellbasierten Systementwurf

Zur Umsetzung von MBSE werden bestimmte Werkzeuge benötigt, wie unter anderem das *Conceptual Data Model* (CDM). In [9] wird das CDM als Datenmodell für Raumfahrtanwendungen definiert, dabei geht der Begriff des CDM zurück in das frühe Datenbank Design [13]. Diese Datenbanken verwalten dabei das gemeinsame System Modell aller Domänen. Alle Domänen können durch Zugriff auf das System Modell Daten lesen, hinzuzufügen oder ändern. Die Datenbank kann diese Änderungen verfolgen und versionisieren. Abbildung 2.4 zeigt das Prinzip der Nutzung von Datenbanken im MBSE. Die einzelnen Domänen geben ihre Informationen an das System Modell weiter und dieses kann einzelne Versionen des System Modells ablegen. Diese Versionsverwaltung kann später genutzt werden, um Versionen des System Modells miteinander zu vergleichen. [13] Diese Versionsverwaltung wurde unter anderem in der Software

Virtual Satellite genutzt, um Änderungen des Raumfahrzeuges im Laufe der Entwicklung mit Hilfe von Visualisierung hervorzuheben.[31]

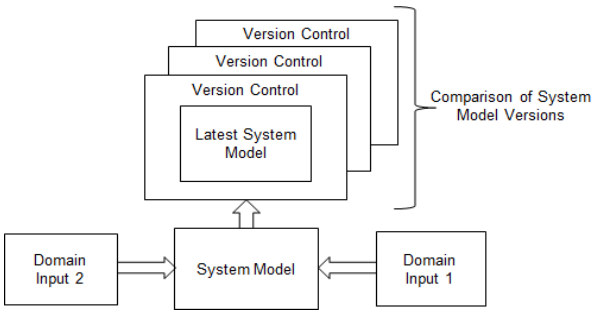

Abbildung 2.4: Datenbanken im MBSE

Ein großer Vorteil der Verwendung von Datenbanken im MBSE ist die Möglichkeit den Informationsaustausch zwischen den einzelnen Domänen konsistent zu halten, somit wird die Erstellung des Systems als einheitliches System verstärkt und Missverständnisse unter den Domänen werden verhindert, sowie Abhängigkeiten früh genug als solche erkannt.

Datenbanken zur Simulatorkonfiguration

Um das Design eines Raumfahrzeuges zu verifizieren und validieren werden in den späteren Lebensphasen einer Raumfahrtmission C und D unter anderem Simulationen durchgeführt. Durch den erst späten Nutzen von Simulatoren kann es vorkommen, dass Fehler oder Änderungen im Design zu spät entdeckt werden und zu kostspieligen Entwurfsänderungen führen können. In [36] wurde eine Studie vorgestellt, in der Simulatoren schon in Phase B des Raumfahrzeugentwurfs eingebunden werden, um somit kostspieligen Änderungen im weiteren Entwurfsprozess zu umgehen. Dabei können Simulatoren ebenfalls von den Informationen des System Modells profitieren. Sind im Laufe des Lebenszyklus des Systems genug Informationen vorhanden, können mittels dieser Simulatoren konfiguriert werden. Ist eine Simulator Architektur erst einmal definiert kann diese genutzt werden um verschiedene spezifische Simulatorenkonfigurationen zu instanziieren.[36]

Für die Thermalsimulation eines Satelliten ist es beispielsweise wichtig zu wissen wie die genaue Struktur des Satelliten, die Komponenten und die Komponentenkonfiguration im Satelliten aussieht.[6] Diese Information kann das System Modell aus dem mechanischen Modell ziehen und an das Thermalsystem weitergeben. Systemtests und Systemsimulationen, wie sie in der Raumfahrt zur Genüge bestehen, basieren auf Strukturinformationen und mechanischen Designinformationen [38], dazu zählen nicht nur Thermal- und Strukturanalysen, sondern auch Vibrationstests und Elektromagnetische-Verträglichkeit-Tests, kurz EMV. All diese Informationen können bei einem konsistenten und einheitlichen System Modell aus diesem entnommen werden.

Kapitel 3

Die MBSE Software Virtual Satellite

Dieses Kapitel stellt den MBSE im Raumfahrzeugentwurf im Deutschen Zentrum für Luft-und Raumfahrt und dabei besonders die Software Virtual Satellite und deren Nutzung zum Aufbau einer Raumfahrtmission vor. Dabei wird die Architektur und Struktur der Software, sowie deren Entwicklungsumgebung näher erläutert.

3.1 Vorstellung und Grundgedanke von Virtual Satellite

Die Software Virtual Satellite ist eine MBSE Datenbank. Bei der Software handelt es sich um ein internes Projekt des Deutschen Zentrums für Luft-und Raumfahrt, welches seit 2010 entwickelt wird. Virtual Satellite soll für den gesamten Entwurfsprozess eines Raumfahrzeugs nutzbar sein. Der Fokus liegt auf der Unterstützung der frühen Entwicklungsphasen und der Concurrent Engineering Facility (CEF) in Bremen. In der CEF wird Virtual Satellite als Werkzeug genutzt, das den Ingenieuren die gemeinschaftliche Arbeit an einer einheitlich digitalen Repräsentation des Systems ermöglicht. [12]

Die Software Virtual Satellite erlaubt den Ingenieuren das Raumfahrzeug in ein digitales Datenmodell zu abstrahieren. Genutzt wird eine hierarchische Top-Down Struktur, die es ermöglicht das Raumfahrzeug vom Gesamtsystem bis hin zu einzelnen Subsystem Komponenten zu zerlegen. Ein *Nutzerrechtmanagment* in der Software sorgt dafür, dass nur der Domänenexperte selbst auf seinem Gebiet Informationen ändern, hinzufügen oder löschen kann. Diese lokalen Änderungen können dann an das sogenannte Version Control System (VCS) weitergegeben werden, welches das aktuelle System Modell mit allen Domäneninformationen beinhaltet. [35] Das untenstehende Schaubild 3.1 erörtert grob das Konzept von Virtual Satellite. Virtual Satellite wird als Datenbank genutzt, in der jeder Domänenbereich seine Änderungen und Informationen ablegen kann. Aus allen Domäneninformationen zusammen erstellt Virtual Satellite dann ein aktuelles, konsistentes, einheitliches System Modell. Das aktuelle System Modell kann dann im VCS hinterlegt werden. [13]

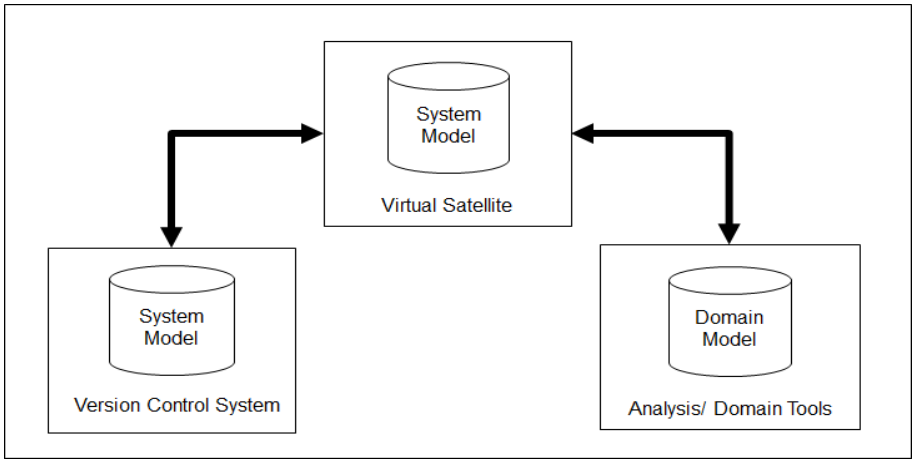

Abbildung 3.1: Prinzip Virtual Satellite

3.1.1 Konzeptdaten Modell und System Modell

Der modellbasierter Systementwurf nutzt bestimmte Werkzeuge, wie Datenbanken um System Informationen zu verwalten. Um die semantische Korrektheit der gespeicherten Informationen sicherzustellen, stellen diese Datenbanken *Conceptual Data Models* (CDM), zu Deutsch Konzeptdaten Modelle, bereit, die in der Informatik oft als Metamodelle bezeichnet werden. Das CDM stellt allen, am Raumfahrzeugentwurf Beteiligten, eine gemeinsame Sprache zur Systembeschreibung zur Verfügung. Das CDM von Virtual Satellite basiert auf dem ECSS-E-TM-10-25 Standard der europäischen Raumfahrtagentur. [34] Das folgende Abbild 3.2 soll die Beziehung zwischen System Modell und Konzeptdaten Modell verdeutlichen. Die Darstellung ist an den System Modeling Language Style angelehnt. Das hier dargestellte CDM bietet eine Sprache zum Definieren von Komponenten. Diese Komponenten können Schnittstellen-Enden aufweisen, um sie mit anderen Komponenten zu verbinden. Das CDM gibt des Weiteren auch Schnittstellen an, die keine vorhandenen Interface-Enden enthalten, sondern als deren Eingang von einer Komponente und deren Ausgang auf eine Komponente dienen. Das dargestellte System Modell bedient sich nun der Sprache und beschreibt mit dieser vier Komponenten, einen Onboard Computer (OBC) und drei Reaktionsräder(RW). Zusätzlich instanziieren alle Komponenten Interface-Enden, die als kleine Kästchen dargestellt sind. Schließlich werden die Schnittstellen-Enden miteinander verbunden, wie am Beispiel der Verbindung zwischen OBC und RW1 zu sehen. Während das Konzeptdaten Modell, nur das *Konzept* enthält, macht das System Modell *Gebrauch* von dessen Konzept.[14]

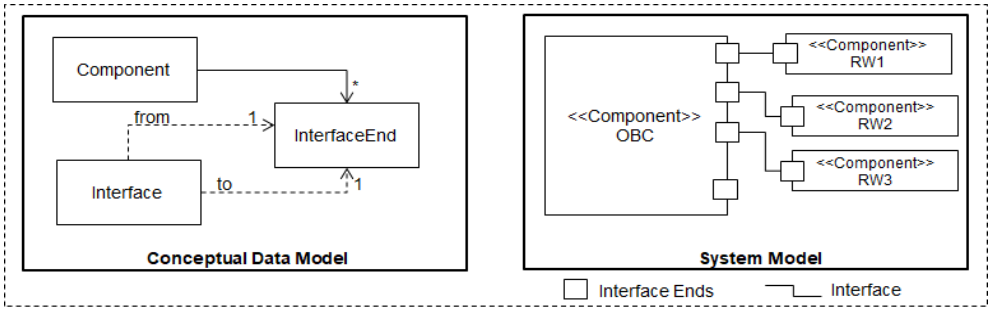

Abbildung 3.2: Relation System Modell und Konzeptdaten Modell[14]

Im Raumfahrzeugentwurf kommen je nach aktueller Lebenslaufphase unterschiedliche Anwendungsfälle zum Tragen. Während Phase A der Verkabelung im Raumfahrzeug wenig Beachtung schenkt, wird diese Information in den späteren Phasen C und D, bei der Integration, auf jeden Fall benötigt. Der Entwurf eines Raumfahrzeugs kann sich über mehrere Jahre hinweg ziehen, so dass es zu Beginn schwierig ist, alle nötigen Anwendungsfälle in der Datenbank von Anfang an unterzubringen. Aus diesem Grund bietet Virtual Satellite die Möglichkeit das CDM je nach Bedarf zu erweitern. Die Abbildung 3.3 beschreibt die Architektur und das Prinzip hinter der Erweiterung in verschiedenen Ebenen des Virtual Satellite Datenmodells. Die unterste Ebene basiert auf Eclipse *Ecore*. Das Ecore-Modell enthält grundlegende Konzepte der objektorientierten Modellierung wie etwa Pakete, Klassen, Referenzen und Attribute. Über der Ecore-Ebene befindet sich die *Generic System Engineering Language* (GSEL), diese beinhaltet Modellierungsfunktionen, die für das System-Engineering und für die Beschreibung zukünftiger Anwendungsfälle des CDM benötigt werden. Über dieser Ebene befindet sich eine weitere Ebene mit drei Konzepten (A, B, C), die Gebrauch machen von der Generic System Engineering Language. Die Konzepte zusammen mit der Generic System Engineering Language bilden das komplette CDM. In der obersten Ebene der Architektur sind die drei Konzepte des CDM in verschiedenen Kombinationen zu sehen. Diese Kombinationen nutzen die Konzepte des CDM um das System Modell zu instanziieren.[14]

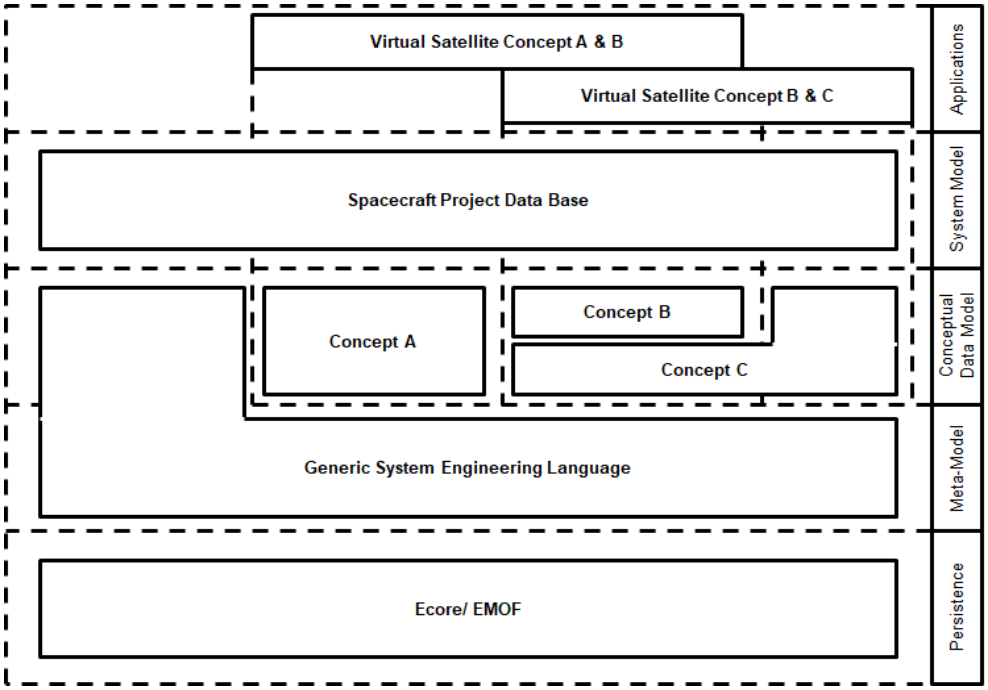

Abbildung 3.3: Ebenen des Virtual Satellite Datenmodells[14]

3.1.2 Konzeptdaten Modell Erweiterung - Engineering Categories

Die GSEL bietet unter anderem auch die Möglichkeit sogenannte *Engineering Categories* zu nutzen. Engineering Categories sind eine Art raumfahrtbezogener Umsetzungen des *Type Object Pattern* und des *Dynamic Template Pattern.*[13] [28] Sie ermöglichen die Definition einer Reihe von Parametern innerhalb einer sogenannte Kategorie, auch als Eigenschaften bezeichnet. In dieser Kategorie werden die Parameter durch einen Namen, aber keinen festen Wert definiert. Da die Engineering Categories innerhalb der GSEL definiert werden können, können diese auch in Konzepten verwendet werden. In dieser Arbeit spielen das Konzept der Produktstruktur und das Konzept der Visualisierung eine tragende Rolle.

Konzept Visualisierung

Das Konzept der Visualisierung besteht aus einer einzigen Category der *Visualization.* Das Konzept wird verwendet um ein dreidimensionales Erscheinungsbild einer Komponente bzw. eines Bauteils zu erstellen. Die Komponenten erhalten durch das Konzept bestimmte geometrische Parameter um die Komponente geometrisch beschreiben zu können. Darunter fallen Parameter wie Form, Farbe und Größe der Komponente, aber auch Parameter die Auskunft über die Lage des Bauteils in der Satellitenkonfiguration geben, wie Rotation und Position. Sollen diese Parameter an Bauteile des Satelliten übergeben werden, zum Beispiel an ein Reaktionsrad, geschieht dies durch ein sogenanntes *Category Assignment*(CA). Dieses CA beinhaltet auch den Wert des Parameters, beispielsweise für die Form den Wert *Box.* Die Kategorie kann

dabei als eine Art Vorlage verstanden werden, welche an einer Komponente instanziiert wird, wenn diese dort benötigt wird. [14] Das nachstehende Schaubild zeigt ein Lageregelungssystem (AOCS - Attitude and Orbit Control) mit zwei Komponenten: einem Reaktionsrad und einem Magnetorquer. Beide Bauteile erhalten nun durch den Gebrauch der Engineering Category *Visualization* eine Form, eine Position in X-Richtung, sowie eine Rotation um die X-Achse.

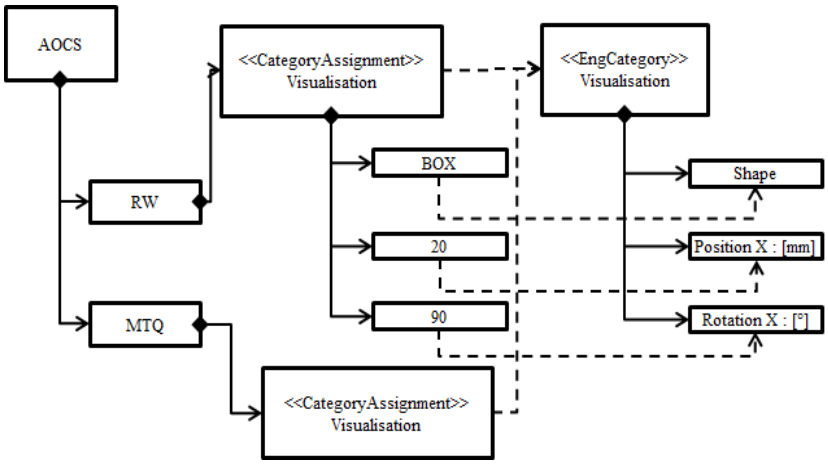

Abbildung 3.4: Visualisierungskonzept in Virtual Satellite

Die Werte der geometrischen Parameter werden durch Category Assignments (vgl. 3.1.2) übergeben und beinhalten für den Parameter der Form eine Box, für die Position in X-Richtung eine Verschiebung um 20mm und für die Rotation um die X-Achse eine Drehung um 90 Grad. Die in Virtual Satellite verwendeten geometrischen Formen zum Darstellen von Bauteilen beziehen sich auf Primitive. Darunter fallen Kugeln, Boxen, Zylinder und Kegel. In den CEF Studien in Bremen, in denen Virtual Satellite genutzt wird, werden nur grobe Abschätzung der Bauteile benötigt und keine detaillierten Information dieser, daher reichen die primitiven geometrischen Formbeschreibungen aus.[51] Die primitiven Formen können anhand von Längen und Radien definiert werden. Für eine detailliertere Darstellung besteht auch die Möglichkeit eine STL-Datei an das Bauteil anzuhängen. Mit Hilfe eines *Visualization Toolkit*, kurz VTK, ist es möglich die Visualisierungsparameter in der Konfiguration direkt zu visualisieren. Die Abbildung 3.5 zeigt beispielhaft die genutzten primitiven Formen zur Visualisierung von Bauteilen. Des Weiteren befinden sich im unteren Bereich der Abbildung zusätzlich die Visualisierungsparameter für das Bauteil, welches als blaue Box in der grafischen Benutzeroberfläche von Virtual Satellite visualisiert wird. Dort sind die Parameter für die Längen der Box, die Farbe, aber auch die Positionen und Rotation in der Beispielkonfiguration dargestellt.

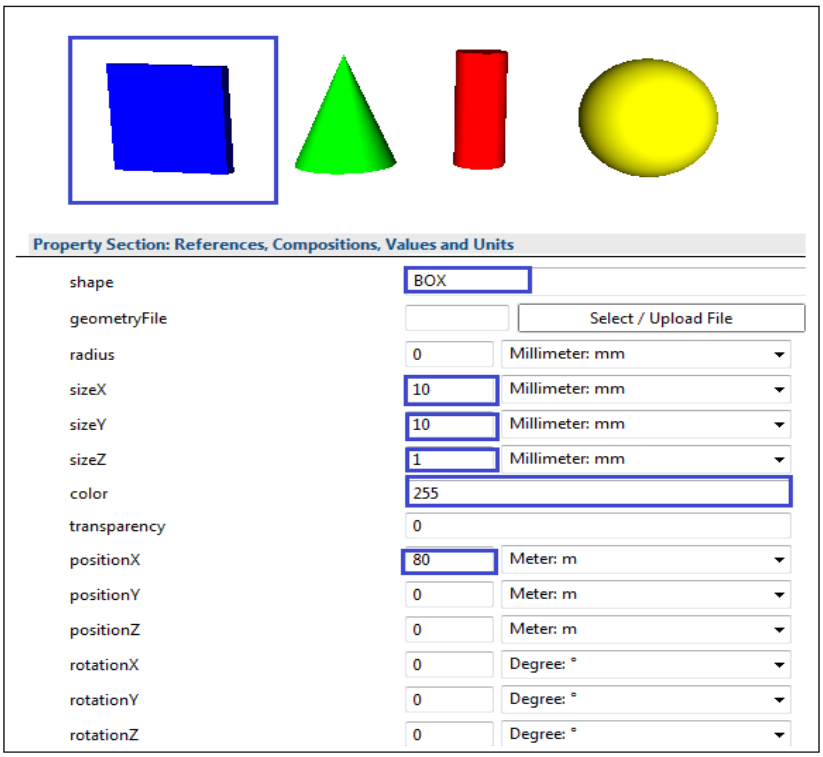

Abbildung 3.5: Primitive geometrische Formen in Virtual Satellite

Konzept Produktstruktur

Die Konzeptdaten Modelle des ECSS Standards enthalten verschiedene Konzepte um die Modellierung eines Raumfahrzeugs zu ermöglichen. Eines dieser Konzepte ist eine baumartige hierarchische Zerlegung des Raumfahrzeugs. Dieses Konzept realisiert die Idee des Organisierens von Informationen in logische Blöcke wie System, Subsystem oder einzelne Komponenten und kann Informationen miteinander in Relation setzen.[11] Beispielsweise stellt eine Menge von Komponenten einen Teil eines Subsystems dar. Ebenso repräsentiert dieses Subsystem einen Teil des Gesamtsystems. Solch eine hierarchische Zerlegung wird oft auch als Produktstruktur bezeichnet. Für die Modellierung eines Raumfahrzeugs wird eine solche Zerlegung des Systems genutzt, um dessen Komplexität zu reduzieren.[29]

Die Software Virtual Satellite bietet die Möglichkeit die Satellitenkonfiguration mittels hierarchischen Baumstrukturen darzustellen, welche auf den ECSS Standards basieren. Dabei unterscheidet man folgende Baumkonzepte: den Product (PT), den Configuration (CT), den Assembly (AT) und den Storage Tree (ST). Abbildung 3.6 zeigt die einzelnen Bäume und die Verbindungen zwischen den Bäumen.

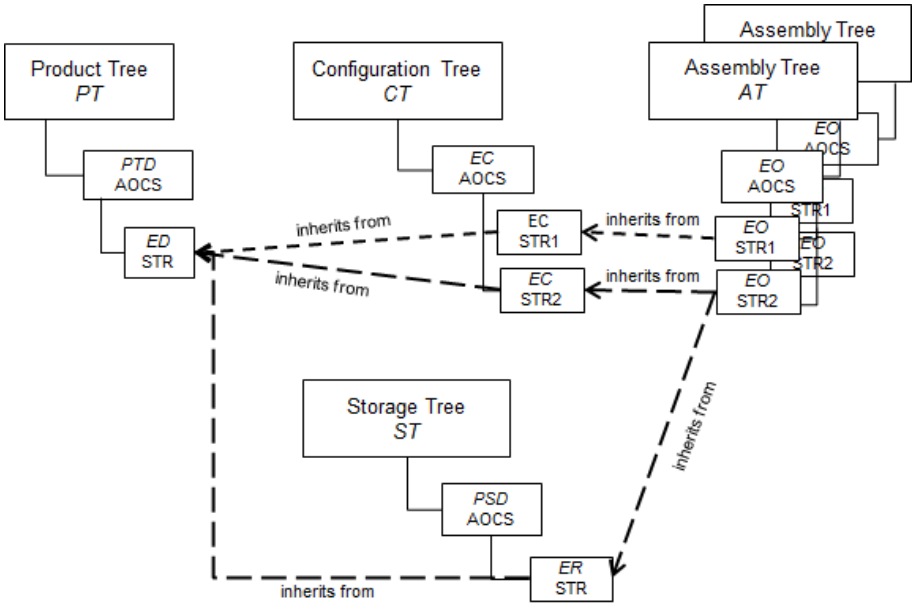

Abbildung 3.6: Die verschiedenen Tree-Konzepte in Virtual Satellite

Im Product Tree werden alle im Raumfahrzeug verwendeten Bauteile abgelegt. Der Product Tree kann dabei Product Tree Domains (PTD) oder Element Definitions (ED) anlegen. Die PTD können in diesem Fall zum Beispiel die einzelnen Subsysteme/ Domänen eines Raumfahrtzeuges darstellen. Die spezifischen Bauteile der Domänen können dann als Element Definition an die Produktbaumdomäne angefügt werden. (vgl. Abbildung 3.6)

Im Configuration Tree werden sogenannte Element Configuration (EC) angelegt, welche Subsysteme oder Bauteile des Raumfahrzeugs darstellen können. Der wesentliche Unterschied zum Product Tree liegt darin, dass der Configuration Tree, die Bauteileigenschaften aus dem Product Tree erben kann. Wurde beispielsweise im Product Tree das Subsystem AOCS (Attitude and Orbit Control) für die Lageregelung angelegt, können dort spezifische Lageregelungsbauteile als Element Definition abgelegt werden, wie beispielsweise ein Sternensensor (STR). Dieser Sternensensor ist ebenfalls im Configuration Tree im Lageregelungssystem wiederzufinden. (vgl. Abbildung 3.6) Der Sternensensor im Configuration Tree erbt nun die Eigenschaften des abgelegten Sternensensors im Product Tree. Da ein Sternsensor nicht allein genutzt wird, sondern meist in einer Konfiguration von zwei oder drei Sternsensoren, kann so auch ein zweiter Sternensensor im Configuration Tree angelegt werden. Während der Product Tree darstellt welches Equipment im Satelliten bzw. Raumfahrzeug genutzt wird, stellt der Configuration Tree eine erste *Konfiguration* des Raumfahrzeugs dar. Des Weiteren enthält er die Information wie oft und wo am Satelliten, welches Bauteil aus dem Product Tree genutzt wird, während der Product Tree nur die Bauteile selbst beinhaltet, nicht jedoch wie oft sie im Satelliten vorkommen. Zudem können in der Satellitenkonfiguration im Configuration Tree erste Annahmen über mögliche Verkabelungen im Satelliten generiert werden.

Der Assembly Tree ist dabei ähnlich zum Configuration Tree aufgebaut, beinhaltet aber die wirkliche und finale Satellitenkonfiguration, abgelegt als Element Occurence (EO). Ein weiterer Unterschied zum Configuration Tree besteht darin, dass es mehrere Assembly Trees geben kann. Dies ist zum einen vorteilhaft, wenn es sich bei der Raumfahrtmission um eine Konstellation handelt und die einzelnen Satelliten sich in Equipment und Aufbau gering unterscheiden. Bei einer Satellitenkonstellation, wie der *FireBIRD* Mission vom Deutschen Zentrum für Luft- und Raumfahrt werden zwei Satelliten zur frühen Detektion von Waldbränden genutzt [7]. Dabei unterscheiden sich die Satelliten nur in deren Nutzlast. Während der einer der Satelliten einen speziellen Infrarotsensor an Bord hat um Waldbrände aufzuspüren, ist der andere mit einer hochauflösenden Kamera ausgestattet, um Bilder von den Waldbränden zu erstellen. Zum anderen kann der Assembly Tree aber auch genutzt werden um Simulatoren zu konfigurieren. Da dieser die tatsächliche Satellitenkonstellation darstellt und Verkabelungen und Positionen aller Bauteile beinhaltet. Diese Informationen sind wichtig um Simulationstests durchführen zu können. Simulationen für Thermalanalysen oder Strukturanalysen benötigen Informationen über die mechanische Konfiguration des Satelliten und Antworten auf die Fragen: Wo ist welches Bauteil in meinem Satelliten angebracht und was befindet sich in dessen Umgebung? Besonders in Bezug auf den Thermaltest spielen diese Fragen eine Rolle, da die Wärmekapazität der Bauteile Auswirkungen auf den Wärmehaushalt des Satelliten hat.

Eine weitere Baumstruktur ist der Storage Tree. Dieser ist ähnlich zum Product Tree aufgebaut beinhaltet aber die tatsächlich genutzten Bauteile, welche im Satelliten integriert werden. Diese werden als Element Realisation (ER) angelegt. Der Storage Tree kann als eine Art digitales Lager gesehen werden, in dem die tatsächlichen Bauteile abgelegt werden. Dabei können für ein Bauteil gleich mehrere Exemplare abgelegt werden. Da ein geliefertes Bauteil Spezifikationen wie Kalibrationskurven und Messwerte enthält ist es sinnvoll mehrere Exemplare zu ordern um diese zu testen. Bei den Spezifikationen des Lieferanten handelt es sich um Wertebereiche, nicht um exakte Werte. Gewählt wird das Bauteil, das den Anforderungen des Systems am besten entspricht. Dieses wird im Satelliten integriert.

3.2 Die Entwicklungsumgebung von Virtual Satellite

Virtual Satellite wird in der objektorientierten Programmiersprache Java 8 entwickelt, basiert auf dem Eclipse Framework und wird auf Basis der Rich Client Plattform (RCP) aufgebaut. Eclipse ist ein quelloffenes Programmierwerkzeug zur Entwicklung von Software und basiert auf dem OSGi-Framework Equinox. Das Equinox Framework ist ein von der Eclipse Foundation entwickeltes Java-basiertes Framework, das OSGi-Kernspezifikation implementiert und die Basis der integrierten Entwicklungsumgebung (IDE) Eclipse bildet. [8] Die Eclipse RCP stellt das Basisgerüst zur Entwicklung von PlugIn-basierten Applikationen dar. Intern besitzt Virtual Satellite einen modularen aus PlugIns bestehenden Aufbau. PlugIns stellen Software-Module dar, die ähnlich eines Baukastenprinzps arbeiten. Sie lassen sich leicht mit anderen Modulen kombinieren. Somit entsteht eine flexible Erweiterung der Applikationen. [46]

Die Eclipse IDE wird von Virtual Satellite aufgebaut und durch PlugIns erweitert, die es ermöglichen ein

Raumfahrzeug in Virtual Satellite aufzusetzen. Virtual Satellite lässt sich durch seine PlugIn-Struktur und das dort enthaltene flexible Datenmodell an unterschiedliche Anforderungen der individuellen Ingenieursaufgaben und Projektanforderungen leicht erweitern und anpassen. Dabei kann das Datenmodell durch sogenannte *Konzepte* erweitert werden. Ein Konzept stellt eine Erweiterung des Grunddatenmodells und deren entsprechenden Funktionalitäten dar, sowie einem User Interface. Dabei gibt es momentan drei Arten von Konzepten in denen Virtual Satellite zur Verfügung gestellt wird, die als folgende Produktlinien aufgefasst werden können.

1. Virtual Satellite Core
2. Virtual Satellite 4 Research
3. Powered by Virtual Satellite 4

Die Produktlinie *Core* beinhaltet dabei die Kernanwendung und beinhaltet alle benötigten Anwendungen um den Entwurf eines Raumfahrtsystems zu beginnen. Erweiterte Funktionalitäten können über Konzepte oder Entwicklungswerkzeuge hinzugefügt werden.

Virtual Satellite 4 Research basiert auf der Core-Produktlinie und bietet individuellen Anwendungen, Experimenten und deren wissenschaftliche Erträge die Entwicklungsmöglichkeit. Das Konzept ist dabei abgegrenzt von der produktiven Software und erlaubt Fehler, die in der produktiven Software so nicht erlaubt wären.

Powered by Virtual Satellite 4, sowie andere Erweiterung von der MBSE Software Virtual Satellite sind maßgeschneiderte Anwendungen für diverse Projekte. Dazu gehört auch das DLR-interne Projekt S2tep. In diesen Projekten werden Funktionalitäten implementiert, die speziell auf die Anforderungen der Projekte eingehen.[12]

Die Eclipse Entwicklungsumgebung bietet die Verwendung des Git Versionskontrollsystem durch das EGit Projekt durch die Einbindung bestimmter PlugIns an. Durch die Verwendung des Git Versionskontrollsystems ist es möglich, dass mehrere Personen gleichzeitig am Quellcode von Virtual Satellite arbeiten und diesen iterativ entwickeln können. In der Projektgruppe des Virtual Satellite finden wöchentliche Treffen statt, um den derzeitigen Implementierungsstand der Software mit allen Mitglieder zu teilen, neue Ideen zu diskutieren und Hilfestellung bei aufgekommenen Problemen zu stellen. Des Weiteren wird die Software testgetrieben anhand von Unit Tests entwickelt. Unit Tests sind Test, die in der Softwareentwicklung angewendet werden, um funktionale Einzelteile (Units) von Programmen zu testen und deren Funktionaliät zu prüfen. [53] Um Unit Tests für Virtual Satellite schreiben und ausführen zu können, wird das Java-Framework JUnit genutzt. Dabei wird zuerst ein automatisch wiederholbarer (JUnit-)Test und dann der zu testende Code geschrieben. Der Test ist genauso ein Stück Software, wie der zu testende Code und wird ebenso programmiert. Beim Durchlaufen der JUnit Tests kann der Programmierer sehen, ob die Abschnitte des zu testenden Codes funktionieren wie gewünscht. Die Idee dabei ist, fehlerarmen

Code zu erzeugen, indem nichts implementiert wird, das nicht auch getestet wird. Werden Testfälle erst nach dem Code entwickelt, so ist die Wahrscheinlichkeit höher, wichtige Testfälle zu übersehen.

Mit Hilfe des *Build-Management Tools*, Maven, kann die Software kompiliert und lauffähig gemacht werden. Durch die *Continous-Integration* Applikation Jenkins werden alle für die Software notwendigen Bibliotheken und Frameworks geladen, die Software gebaut und alle Tests durchlaufen. Der Entwickler kann dann nach Beendigung des *Softwarebaus* die aktuelle Versionsstabilität prüfen. Der im Zuge dieser Arbeit implementierte Quellcode wurde in die Virtual Satellite Research Produktlinie eingepflegt, welche die Virtual Satellite IDE nutzt, und unter JUnit Tests geprüft.

Kapitel 4

Integration von mechanischem Design in den MBSE

Das folgende Kapitel stellt das mechanische Design und die Bedeutung von CAD Software im mechanischen Entwurf einer Raumfahrtmission vor. Im Speziellen wird hier auf die CAD Software CATIA eingegangen, die in dieser Arbeit für die Integration von mechanischem Design in den MBSE genutzt wurde. Darauffolgend werden einige Konzepte vorgestellt, die die Integration von mechanischem Design in den MBSE unidirektional bereits vorgenommen haben. Des Weiteren werden bestehende Modelltransformationen und Austauschformate vorgestellt.

4.1 Mechanisches Design

Als mechanisches Design ist die Konstruktion und der mechanische Entwurf eines Systems, sowie dessen Analyse gemeint. Im mechanischen Design können einzelne Bauteilkomponenten des Systems unter Berücksichtigung von geeigneten Dimensionierungen, Formen und Materialien konstruiert werden. Des Weiteren können die Bauteilkomponenten als Zusammenbau, auch oft Assembly genannt, integriert werden. Zur Unterstützung des mechanischen Designs werden CAD (Computer-Aided-Design) Werkzeuge genutzt. CAD bezeichnet die Hilfestellung von konstruktiven Aufgaben mittels elektronischer Datenverarbeitung zur Herstellung eines Produkts. Durch die Nutzung von CAD Werkzeugen kann das Verständnis des Produkts bzw. Systems durch die dadurch entstandene CAD Visualisierung gesteigert werden. Eine der bekannteren Werkzeuge im CAD Bereich ist zurzeit CATIA. Auf die Software CATIA wird im Folgenden näher eingegangen.

4.1.1 Die CAD Software CATIA

CATIA steht für *Computer Aided Three-Dimensional Interactive Application* und ist ein CAD-Tool der französischen Firma Dassault Systèmes. Ursprünglich für den Flugzeugbau entwickelt hat sich das CAD Tool heute in verschiedenen Branchen etabliert, wie auch in der Raumfahrtbranche.[48] Es bietet die Möglichkeit einzelne Bauteile zu entwerfen und diese in größeren Baugruppen zu integrieren und wiederzuverwenden. Dabei werden die Bauteile in sogenannten *CATParts* gespeichert und die Baugruppen in einem *Assembly (CATProduct)*. Dabei können den CATParts verschiedene mechanische Eigenschaften,

wie Masse, Material, Dichte, thermische Leitfähigkeit etc. zugeorndet werden. So kann auch ein Satellit erst in einfache Bauteile zerlegt werden, bevor man den gesamten komplexen Satelliten in einem Assembly als Konfiguration entwirft. Die nachfolgende Abbildung soll die in CATIA genutzte Produktstruktur bezüglich eines Assemblies und dessen Zusammenstellung verdeutlichen, die Dekomposition ist im linken Abschnitt der Abbildung zu sehen.

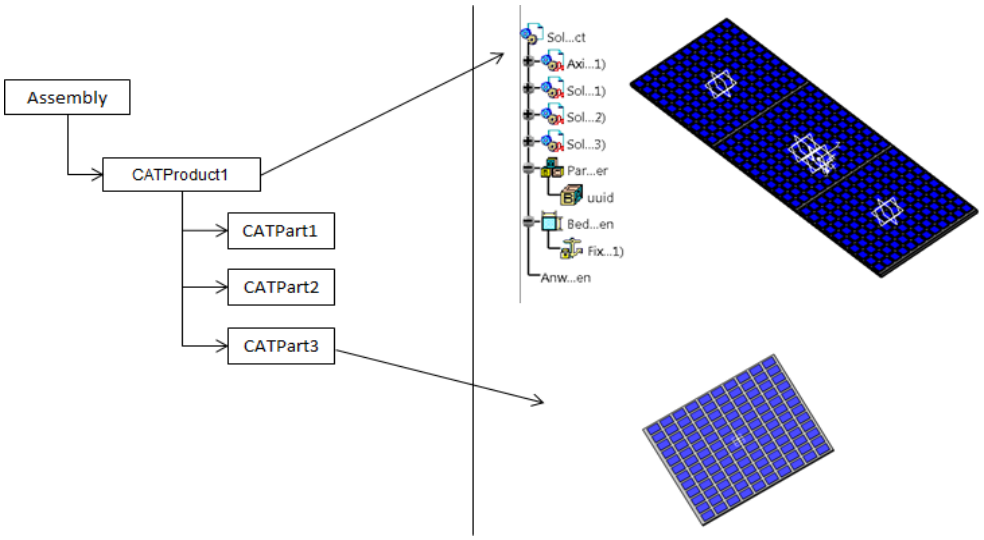

Abbildung 4.1: CATIA Assembly - Hierarchische Baumstruktur

Das gezeigte Assembly besteht aus einem CATProduct, welches den Zusammenbau von drei Solarpanelen beschreibt. Jedes Solarpanel ist dabei ein CATPart. Mit Hilfe von mechanischen Bedingungen (CATIA Constraints) können die Bauteile der Baugruppe miteinander verknüpft werden. In diesem Beispiel wurden die einzelnen CATParts der Solarpanele mittels Flächenkontaktbedingungen zu einer Baugruppe (CATProduct) zusammen gebaut.

4.1.2 Bedeutung von mechanischem Design in Raumfahrtmissionen

Durch das mechanische Design kann der Satellit bzw. das Raumfahrzeug als Ganzes mit Verbindungen der einzelnen Komponenten als mechanische Konfiguration dargestellt werden. Durch CAD Tools kann dieses mechanische Design in frühen Entwurfsphasen bereits visualisiert werden, um das Verständnis der gesamten mechanischen Konfiguration zu verbessern und nach und nach weiter zu detaillieren. Bereits in den frühen Entwurfsphasen tritt die Frage auf, ob der Satellit in die Nutzlastverkleidung der Rakete (Fairing) passt. In den späteren Entwurfsphasen spielt die mechanische Konfiguration für weiterführende Simulationen, wie die Thermal- oder Strukturanalyse eine wichtige Rolle. In Raumfahrzeugmissionen

wird der Entwurf in Phasen unterteilt. Durch die geometrische Darstellung des Satelliten durch die CAD Software können dabei geometrischen Anforderungen des Systems verifiziert und über den kompletten Entwurf der Mission in den einzelnen Phasen verfolgt werden. Handelt es sich beispielsweise um eine CubeSat Mission stehen Größe und Form des Satelliten schon fest und ein erstes Modell des Satelliten kann mittels CAD Software erzeugt werden und die geometrischen Anforderungen können visualisiert werden.[32]

4.2 Stand der Technik

Die Kernidee der Kopplung des mechanischen Designs an den modellbasierten Systementwurf ist nicht neu. Es hat in diesem Bereich schon verschiedene Umsetzungen gegeben. Jedoch sind diese Kopplungen nur unidirektional, was bedeutet, dass entweder nur CAD Modelle aus dem System Modell erzeugt oder die Informationen aus dem CAD Modell in das System Modell importiert werden können. Einige Beispiele dieser Umsetzungen, sowie technische Methoden zum Austausch zwischen MBSE und mechanischen Design werden nachfolgend vorgestellt.

4.2.1 Einsatz von CAD Modellen in den Raumfahrtdomänen

Durch CAD Tools wie CATIA ist es möglich, dass alle Domänen ein besseres Verständnis bezüglich der Gesamtkonfiguration des Satelliten bekommen und Unstimmigkeiten unter den Domänen schneller entdeckt werden können. Schaut sich ein Thermalingenieur den mechanischen Aufbau des Satelliten an und findet er dort aufgrund der Materialien der Komponenten Unstimmigkeiten die Auswirkungen auf den Thermalhaushalt des Satelliten haben, kann er dieses aus dem CAD Modell direkt entnehmen. Auch der Lageregelungsingenieur kann mit Angaben bezüglich Parametern, wie Schwerpunkt und Trägheit, welche in CATIA direkt berechnet werden können, Aussagen zur Satelliten Lageregelung treffen und sein Subsystem damit verifizieren und gegebenenfalls Änderungen vornehmen oder Änderungen über das System Modell an das mechanische Modell weitergeben. Da CAD Modelle sehr detaillierte Beschreibungen des Systems enthalten können sie als wichtige Informationsquelle in der Simulatorkonfiguration dienen. In [44] wurde die Simulatorkonfiguration eines Flugzeugflügels anhand eines CAD Modelles vorgenommen. Dabei wurden aufbauend auf dem CAD Modell die exakten geometrischen Positionen und Parameter in eine Datei exportiert, welche anschließend an das Simulations Tool SimMechanics weitergegeben wurde um dort die Simulationen auszuführen.

Auch in der Raumfahrt können CAD Modelle als wichtige Informationsquelle im Bereich der Simulatorkonfiguration angesehen werden. Nahezu alle Systemtests können in der Raumfahrt nicht unter *realen* Umgebungsbedingungen durchgeführt werden, da ein Satellit nicht vorher im Weltraum getestet werden kann. Aus diesem Grund gibt es Simulatoren um die Weltraumbedingungen zu simulieren und den Satelliten unter halbwegs realen Bedingungen zu testen. Für die Thermalanalyse eines Satelliten kann zum

Beispiel die Software ESATAN genutzt werden, welche mit Hilfe von einem Modell des Satelliten und dessen Orbit eine Thermalsimulation erstellen kann. [49] Je genauer das Satelliten Modell dabei ist, desto genauer ist die Thermalsimulation. Das mechanische Design Modell stellt dabei zu diesem Zeitpunkt der Raumfahrtmission, in Phase D, ein sehr detailliertes Modell, dar. Auf Grund der sehr detaillierten Beschreibung des Satelliten durch das CAD Modell besteht nicht nur die Möglichkeit die Thermalsimulation zu unterstützen, sondern auch FEM-, Vibrations- und elektromagnetische Verträglichkeit (EMV) Simulationen, da alle Simulationen auf einem Modell basieren.[18] Je detaillierter dabei das Modell der Simulation ausfällt, desto genauer und präziser ist das Ergebnis dieser.

In [27] wurde gezeigt wie mittels Entwurfssprachen [26] detaillierte Simulationsmodelle erstellt werden können. Die dort genutzte Entwurfssprache wird abstrakt mit Vokabeln und Regeln abgebildet und dann von einem Entwurfscompiler übersetzt.[19] Dabei wird die Entwurfssprache bestehend aus den abstrakten geometrischen Beschreibungen in CAD Tools, wie CATIA oder Opencascade, exportiert, um die geometrischen Beschreibungen in ein CAD Modell umzuwandeln und anschließend aus diesen Simulationsmodelle zu erstellen.

4.2.2 Import von mechanischem Design in ein System Modell

Die Software Virtual Spacecraft Design (VSD) der europäischen Raumfahrtagentur (ESA) bietet die Möglichkeit das mechanische Design eines Raumfahrzeugs aus einem CATIA Modell in das System Modell zu importieren. Dabei wird mit Hilfe eines geeigneten Austauschformats eine Datei erstellt, in der alle relevanten Informationen aus dem CAD Modell enthalten sind, dazu zählen unter anderem Masse, Volumen, Dichte, Schwerpunkt und Massenträgheit einzelner Bauteile. Des Weiteren wurde ein *Static Mapping File* generiert, welches CATIA Eigenschaften, an das zugehörige VSD Bauteil anhängt. Die Zuordnung erfolgt beim ersten Import, dort werden die Bauteilnamen aus dem CATIA Modell den zugehörigen Bauteilen im VSD System Modell manuell durch den Nutzer angehängt. Das Bauteil im VSD Modell besitzt dabei einen sogenannten *Universally Unique Identifier*, kurz UUID. Das Static Mapping File ordnet dabei die Bauteilnamen aus dem CAD Modell, den Bauteil UUIDs aus dem VSD Modell zu.[15] Abbildung 4.2 zeigt den sogenannten CATIA Mapper, der als Wizard zur Verfügung steht, um die CATIA Eigenschaften an das VSD Bauteil anzuhängen. Dabei ist auf der linken Seite das CATIA Bauteil mit den Eigenschaften abgebildet und auf der rechten Seite das Bauteil im VSD System Modell. Das im VSD genutzte Austauschformat ist das json-Format.

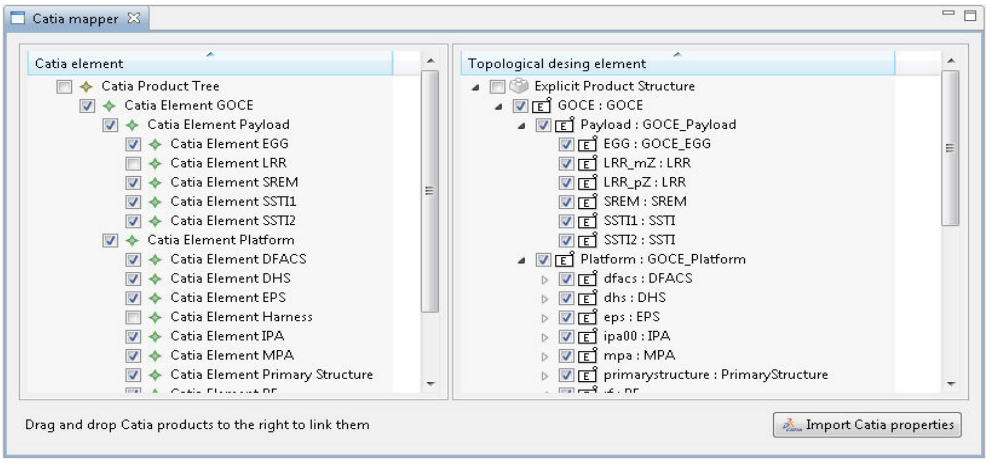

Abbildung 4.2: VSD Import mit CATIA Mapper[15]

Der unidirektionale Weg von mechanischem Modell in das System Modell mag gewisse Vorteile aufzeigen, jedoch zeigt sich deutlich ein Bruch, wenn es um bestimmte Änderungen im System geht. Im Fall von VSD kann das mechanische Modell zwar in das System Modell integriert werden, führt das System Modell nun aber Änderungen durch, können diese nicht in das CAD Modell reimportiert werden. Ein Reimport von mechanischem Modell in das System Modell ist jedoch mit VSD realisierbar.

Ein weiteres Konzept zum Import von mechanischen Design Informationen aus CAD Modellen in das System Modell wurde in [32] vorgenommen. Die von der NASA (National AeronautIcs and Space Agency) im Verlauf der JPL (Jet Propulsion Laboratory) Studie entworfene MBSE Plattform Syndeia ist in der Lage mittels *Drag and Drop* aus dem System Modell Graph ein CAD Modell zu erstellen oder dieses mit dem System Modell zu verknüpfen. Dabei wird beim Import des mechanischen Designs nicht das Modell als solches visualisiert, sondern die System Architektur und die Dekomposition des CAD Modells in das System Modell anhand von UML Diagrammen übertragen.

4.2.3 Export von System Modell in ein mechanisches Modell

Die Version Virtual Satellite 3 stellt einen Export von System Modell in das mechanische Modell bereit, auch hier wurde die CAD Software CATIA genutzt. Dabei wird die Satellitenkonfiguration in Virtual Satellite mit Hilfe der dort vorhandenen Visualisierung erstellt. Die Konfiguration des Satelliten kann dann in ein Script exportiert werden, welches anschließend in CATIA eingelesen werden kann. Das erzeugte Script basiert auf der CATIA eigenen Sprache, des CATScripts. Es ist eine auf Visual Basic basierende Scriptsprache und ermöglicht aktives Scripting in Catia V5. Die einzelnen erstellten Komponenten aus

Virtual Satellite werden dann wie in der dort erstellten visuellen Konfiguration positioniert und in CATIA als Assembly erstellt.

Das Script beinhaltet Daten über Hierarchie, Farbe, Form, Orientierung und Position der einzelnen Komponenten relativ zur Satellitenkonfiguration. Der CAD Export aus der MBSE Software ermöglicht den einzelnen Domänen ein verbessertes Verständnis der Satellitenkonfiguration und zudem eine verbesserte Zusammenarbeit an dieser. [42] Die Abbildung 4.3 zeigt den Prozess des Exports von Virtual Satellite in CATIA. Der linke Teil zeigt die Visualisierung des Satelliten in Virtual Satellite, während der rechte Teil der Abbildung den exportierte Satelliten als Assembly in CATIA zeigt. [33]

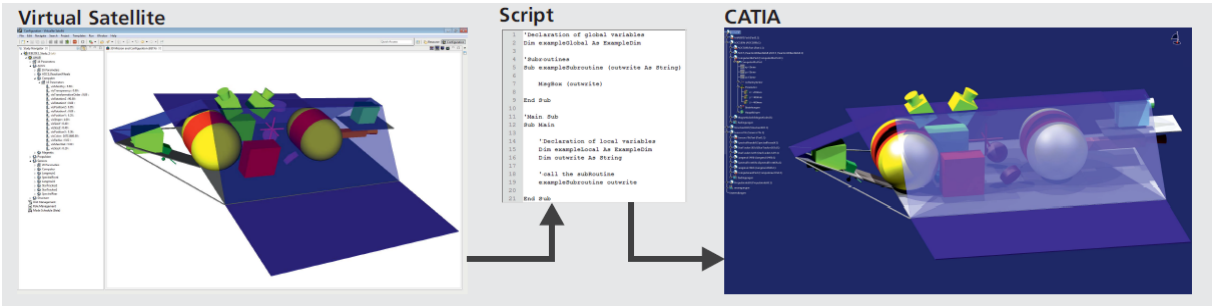

Abbildung 4.3: CAD Export Prinzip Virtual Satellite 3[42]

Der hier gezeigte unidirektionale Weg des Exports zeigt Komplikationen auf, wenn es darum geht, am exportierten CAD Modell Änderungen vorzunehmen, diese können nicht in das System Modell reimportiert werden. Auch eine Aktualisierung des CAD Modells, durch das System Modell ist ausgeschlossen. Bei einem Reexport des System Modells in ein CAD Modell wird im Fall von Virtual Satellite 3 ein komplett neues CAD Modell erstellt, welches abgekoppelt von vorherigen CAD Modellen ist.

4.2.4 Methoden der Modelltransformationen

Um die Modelltransformation zwischen CAD Modell und System Modell vorzunehmen können die Engineering Categories (vgl. Kapitel 3.1.2) genutzt werden, welche raumfahrtbezogene Umsetzungen des Type-Object Patterns darstellen. Diese spielen eine wichtige Rolle beim Übertragen und Wiederverwenden von Informationen. Im Fall der vorgestellten Software Virtual Spacecraft Design werden basierend auf den Engineering Categories in beiden Modellen Kategorien erstellt, welche dieselben Parameter aufweisen, um ein einfaches Mapping auf Konzeptebene zu ermöglichen. Diese Parameter beinhalten Eigenschaften zu Masse, Dichte und anderen mechanischen Informationen, die aus dem CAD Modell gewonnen werden können. In einem zweiten Schritt des Mappings kann die Zuordnung der CAD Informationen im System Modell auf Instanz Ebene erfolgen, in dem die UUID aus dem VSD genutzt wird. Mit dieser ist es

möglich, das Bauteil im System Modell ausfindig zu machen und mit dem Bauteil im CAD Modell zu verknüpfen, um die dortigen mechanischen Informationen im System Modell zu hinterlegen.[13] In [36] wird für die Anbindung von Simulatoren an das System Modell ebenfalls eine Modellzuordnung bzw. ein *Modell Mapping* anhand der Engineering Categories vorgenommen.

[32] nutzt für die Integration von CAD Informationen in der System Modell Plattform den selbst generierten *Connection Manager*. Durch diesen können durch einfache Drag-and-Drop Operationen Verbindungen zwischen CAD und System Modell hergestellt werden. Dabei wird zwischen zwei Verbindungen unterschieden, der *Reference Connection* und der *Model Transform Connection*. Die Reference Connection erstellt eine Verbindung zwischen CAD Element und System Element, während die Model Transform Connection aus dem CAD Modell ein SysML Diagramm erstellt, das der CAD Assembly-Struktur entspricht. Dabei wird für jedes Assembly bzw. Subassembly ein solches SysML Diagramm mit Eigenschaften der einzelnen Komponenten im Assembly angelegt, unter anderem für CAD Massen- und selbstdefinierte Parameter.

4.2.5 Austauschmethoden zwischen MBSE und mechanischem Design

Für den Austausch zwischen MBSE und mechanischem Design bestehen unterschiedliche Methoden und Austauschformate. Im Folgenden werden einige der bestehenden Methoden und Formate explizit vorgestellt.

4.2.5.1 Remote Procedure Call

Eine betrachtete Methode beschreibt den Remote Procedure Call (RPC). RPC ist eine Möglichkeit, ein Client-Server-Modell zu implementieren. In diesem Modell schickt der Client eine Anfrage an einen bekannten Server und wartet auf dessen Antwort. Der Client gibt dabei in der Anfrage an, welche Funktion er mit welchen Parametern ausgeführt haben möchte. Die Anfrage wird dann vom Server bearbeitet und dieser schickt die Antwort zurück an den Client. Sobald der Client die Antwort empfangen hat, kann er seine Verarbeitung weiterführen. Abbildung 4.4 zeigt den Ablauf eines RPC-Aufrufs. Im ersten Schritt führt die Client-Anwendung einen Prozeduraufruf durch, wie sie auch einen lokalen Prozeduraufruf ausführen würde. Die Prozedur, die vom Client aufgerufen wird, wird als *Stub*, deutsch: Kontrollabschnitt, bezeichnet. Der Stub verpackt die zu übergebenen Argumente mit zusätzlichen Informationen, die der Server benötigt, und leitet dieses gesamte Paket an die Client-Netzwerkschnittstelle weiter.[40]

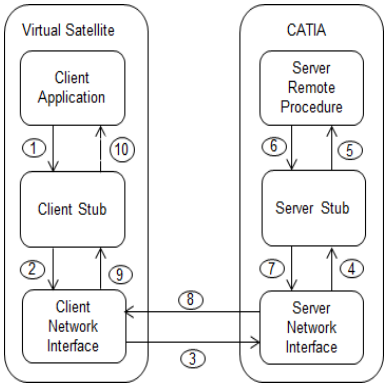

Abbildung 4.4: Prinzip RPC [40]

4.2.5.2 Native CAD Austauschformate

CAD Tools stellen die Möglichkeit bereit, über native Austauschformate Informationen, insbesondere mit anderen CAD Tools, auszutauschen. Eines der bekannteren Austauschformate, besonders in Bezug auf CAD Modelle, ist das ISO-Format STEP. Dieses ist direkt generierbar und benötigt keine weiteren Eingriffe in die Software. Ein weiteres direkt ableitbares Format ist STL. Beide Formate werden im Folgenden vorgestellt.

STEP-Format

STEP steht für *STandard for the Exchange of Product model Data* und ist über die ISO-Norm 10303 definiert. Das Format geht über den reinen Geometriedatenaustausch hinaus und ist aufgrund seiner Standardisierung durch die ISO-Norm besonders für den Datenaustausch zwischen unterschiedlichen CAx Systemen geeignet.[16] Innerhalb des STEP Formats können Produktdaten Informationen des gesamten Lebenszyklus abgebildet werden. Hierfür werden sogenannte Applikationsprotokolle für die unterschiedliche Anwendungsbereiche zur Verfügung gestellt.[10] Für die Geometriebeschreibung mit STEP können sämtliche Formen von CAD-Datenmodellen integriert werden, dazu zählen unter anderem Flächen-, Draht und Volumenmodelle.[2]

STL-Format

STL (Standard Tesselation Language) ist ein Datenaustauschformat vieler CAD Systeme. Dabei werden mittels Dreiecksfacetten 3D Körper beschrieben. Das Format beschreibt allerdings nur die Oberfläche eines Körpers.[43] Abbildung 4.5 zeigt die Darstellung eines hohlen CAD Zylinders mit Hilfe der genutzten STL Dreickecksfacetten im Vergleich zu dem herkömmlichen CAD Modell. Bei gekrümmten Oberflächen bietet STL nur eine Näherung durch die Dreiecke. Um die Genauigkeit der Näherung zu erhöhen werden viele kleine Dreiecksfacetten benötigt.

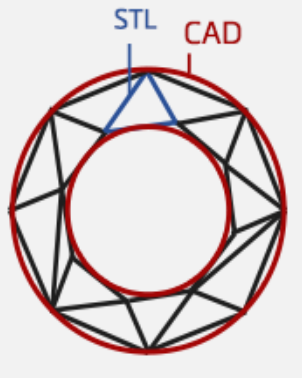

Abbildung 4.5: STL vs. CAD [4]

4.2.5.3 CATIA Visual Basic Application

Das CAD Tool CATIA bietet die Möglichkeit mit einer Schnittstelle zur Visual Basic Application (CATIA-VBA), weitere Formate für den Austausch in Betracht zu ziehen, die keine nativen CAD Formate darstellen. VBA ist eine Skriptsprache für die Steuerung von Abläufen der Microsoft-Office-Programmfamilie. Sie wurde aus dem von Microsoft entwickelten BASIC-Dialekt Visual Basic (VB) abgeleitet und wird in CATIA als Makrosprache genutzt. Dabei werden die Makros in ein CATScript geschrieben. Ein Makro ist in der Softwareentwicklung eine unter einer bestimmten Bezeichnung (Makroname) zusammengefasste Folge von Anweisungen oder Deklarationen, um diese mit nur einem einfachen Aufruf ausführen zu können. Alle Anweisungen des Makros werden automatisch an der Stelle im Programm ausgeführt, an denen das Makro codiert wurde. [3] Hier wurden das json-Format und das XML-Format betrachtet. Beide Formate stellen keine nativen CAD-Formate dar.

XML-Format

Die Erweiterbare Auszeichnungssprache (englisch *Extensible Markup Language*), abgekürzt XML, ist ein textbasiertes Austauschformat, das für den Austausch strukturierter Informationen verwendet wird. Bei XML handelt es sich um ein plattform- und implementationsunabhängiges Austauschformat, das insbesondere für den Austausch von Daten zwischen Computersystemen über das Internet genutzt wird. XML wurde 1998 vom World Wide Web Consortium (W3C) veröffentlicht.[41] Eine wichtige Datenstruktur im XML-Format ist das Element. Elemente besitzen Text oder auch weitere Elemente als Inhalt. Sie bilden die Knoten des Strukturbaumes eines XML-Dokumentes und transportieren die Informationen. Mit dem Format lassen sich hierarchische Baumstrukturen sehr gut darstellen, da durch die Struktureinheit des Elements eine leichte Verschachtlung von Informationen möglich wird. [17] Abbildung 4.6 zeigt ein einfaches Beispiel zur Definition eines Objekts mit Hilfe von XML Elementen. Das Objekt *person* besitzt Informationen über die Person Max Mustermann.

```
<person>

    <vorname>Max</vorname>
    <nachname>Mustermann</nachname>
    <alter>30</alter>
    <kinder>Kind1</kinder>
    <kinder>Kind2</kinder>

</person>
```

Abbildung 4.6: Ein einfaches Objekt im XML Format

Json-Format

Json (JavaScript Object Notation) ist ein schlankes Datenaustauschformat. Es basiert auf einer Untermenge der JavaScript Programmiersprache. [23] Bei json handelt es sich ebenfalls um ein Textformat, das komplett unabhängig von Programmiersprachen ist. Json wird mit zwei primären Datenstrukturen dargestellt: geordnete Listen, auch als *Arrays* (Zeichenketten) bezeichnet, und Name/Wert-Paare, unter anderem unter dem Begriff *Objekt* bekannt. [41] Hierbei handelt es sich um universelle Datenstrukturen, die von so gut wie allen modernen Programmiersprachen in der einen oder anderen Form unterstützt werden.

Abbildung 4.7 zeigt die beiden Datenstrukturen anhand des Beispiels, welches auch für das XML-Format genutzt wurde. Das Objekt enthält dieselben Informationen zur Person Max Mustermann wie im XML Beispiel und wird durch geschweifte Klammern umschlossen. Im Objekt befindet sich zudem eine Zeichenkette, welche die Kinder der Person enthält. Diese sind in einer Zeichenkette deklariert, da es mehrere Kinder gibt. Eine Zeichenkette wird von eckigen Klammern umschlossen. Die Werte innerhalb der Zeichenkette werden durch Kommas voneinander getrennt.

```
{

    "Vorname" : "Max",
    "Nachname": "Mustermann",
    "Alter"   : "30",
    "Kinder"  : ["Kind1", "Kind2"]

}
```

Abbildung 4.7: Das json-Format

Im Array selbst können wiederum Objekte angelegt werden. Die beiden Datenstrukturen können beliebig oft ineinander verschachtelt werden, um so komplexe Datenstrukturen darstellen zu können. Das json-Format wurde in der unidirektionalen Integration von mechanischem Design an das System Modell in der Software Virtual Spacecraft Design bereits erfolgreich implementiert.

Kapitel 5

Entwurf einer bidirektionalen Integration

Dieses Kapitel zeigt den Informationsfluss zwischen mechanischem Design und MBSE und dessen Kriterien für die Bidirektionalität der Integration, nachdem das Ziel der vorliegenden Arbeit dem derzeitigen Stand der Technik gegenübergestellt wird.

5.1 Gegenüberstellung von Ziel und Stand der Technik

Das nachstehende Schaubild 5.1 stellt den Stand der Technik des Ziels der Masterarbeit gegenüber. Im Fokus steht der Übergang von einem unidirektionalem Austausch zu einem bidirektionalen Austausch zwischen MBSE und mechanischem Design. Während Virtual Satellite 3 nur einen Export des System Modells in das mechanische Modell ermöglicht, bietet das Virtual Spacecraft Design die Möglichkeit das mechanische Modell in das System Modell zu importieren. Auch ein Reimport des mechanisches Modells in das System Modell ist mittels VSD möglich. Das komplette Roundtrip Engineering zwischen mechanischem Design und System Modell ist jedoch in den bestehenden Methoden nicht vorhanden. Dabei soll anhand dieser Arbeit eine bidirektionale Schnittstelle erstellt werden, die sowohl den Import und Export, als auch den Reimport und Reexport des System Modells in das mechanische Modell ermöglicht. Dieser Austausch soll über den gesamten Lebenszyklus einer Raumfahrtmission erfolgen, so dass ebenfalls eine Konfiguration von Simulatoren über das mechanische Design in Betracht gezogen werden kann. Anlehnend an die beiden erwähnten Konzepte der ESA und des DLR wird die bidirektionale Schnittstelle dieser Masterarbeit erstellt.

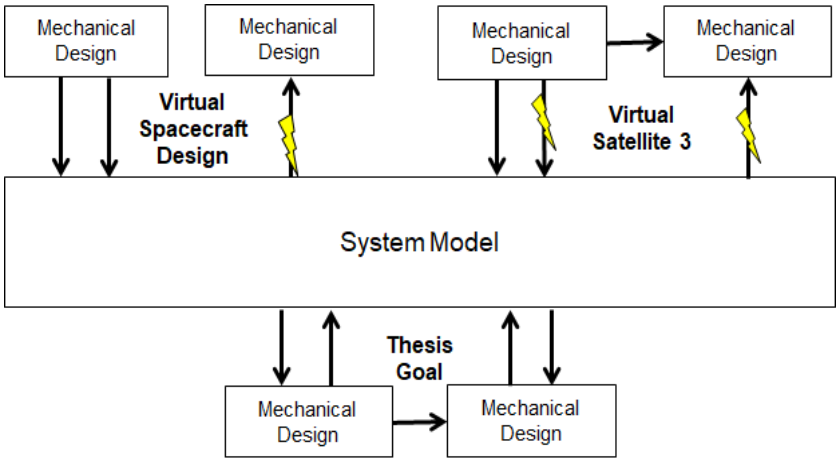

Abbildung 5.1: Stand der Technik vs. Ziel der Masterarbeit

Während die unidirektionalen Konzepte der ESA und des DLR auf Komplikationen stoßen, wenn es um Änderungen in den jeweiligen Modellen und den dortigen Reimport dieser Änderungen in das jeweilig andere Modell geht, soll die bidirektionale Schnittstelle das komplette Roundtrip Engineering ermöglichen. Im folgenden Abschnitt werden Anforderungen an diese bidirektionale Kopplung festgehalten.

5.2 Vorteile des Roundtrip Engineerings

Eine bidirektionale Integration von mechanischem Design in den übergeordneten Prozess des modellbasierten Systementwurfs bietet grundlegende Vorteile in der Entwicklung eines Raumfahrzeugs. Durch die Bidirektionalität entsteht ein Roundtrip Engineering, das es ermöglicht die Informationen zwischen beiden Modellen konsistent zu halten. Dabei können beide Modelle über den Entwurfszyklus einer Raumfahrtmission separat voneinander verfeinert werden, durch das Rountrip Engineering entsteht jedoch die Möglichkeit das System Modell durch das mechanische Modell und das mechanische Modell durch das System Modell zu aktualisieren.

Das System Modell bietet die Möglichkeit den Satelliten als Modell zu visualisieren, dennoch beinhaltet ein mechanisches Designmodell, welches Computer-Aided-Design (CAD) Software nutzt ein viel komplexeres und detailreicheres Modell des Satelliten. Beispielsweise bieten CAD Modelle den erleichterten Zugang zu Massenparametern, Geometrieparametern und anderen relevanten Strukturinformationen, wie Trägheitsmomente, Schwerpunkten und Materialinformationen. Die Informationen können über die Schnittstelle an das System Modell übergeben und übernommen werden. Ein Vorteil der Bidirektionalität ist nun, dass Änderungen, welche im System Modell bezüglich dieser Parameter auftauchen wieder zurück an das mechanische Modell gegeben werden können, wenn diese beispielsweise durch einen Domäneningenieur geändert wurden.

Informationen zu Systemanforderungen die geometrische Anforderungen bereitstellen, wie Größe oder Gewicht eines Satelliten, können über die bidirektionale Schnittstelle an das mechanische Design übergeben und dort im Laufe des Entwurfszyklus ausgetauscht werden. Das bietet die Möglichkeit solche Anforderungen über die kompletten Entwurfsphasen hinweg zu verfolgen und zu verifizieren.

Des Weiteren bieten CAD Modelle eine wichtige Informationsquelle für Systemsimulationen. CAD Modelle eines Raumfahrzeugs stellen detaillierte mechanische Konfigurationen dar, welche in den Simulationen, wie Thermal- oder Strukturanalyse, wiederverwendet werden können. Abbildung 5.2 zeigt den Informationsaustausch zwischen mechanischem Design und der Simulatorkonfiguration über das System Modell für den Fall einer Thermalsimulation. Über die Anbindung des CAD Modells an das übergeordnete System Modell können die detaillierten Informationen über die mechanische Konfiguration an Simulatoren weitergegeben werden. Falls sich im Laufe von Simulationen Ergebnisse zeigen, welche Änderungen in der Konfiguration zur Folge haben, können diese Informationen über das System Modell zurück an das mechanische Design und an das CAD Modell gegeben werden. Somit können auch andere Domänen von der Bidirektionalität der Integration profitieren.

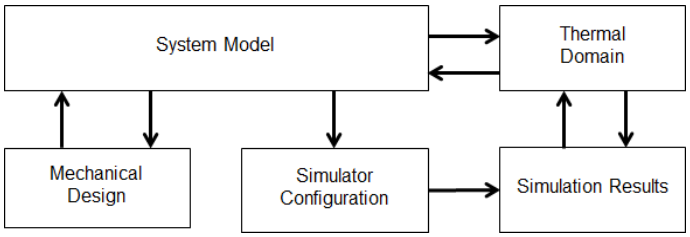

Abbildung 5.2: Simulatorkonfiguration durch CAD Modelle

5.3 Informationsfluss zwischen mechanischem Design und MBSE

Für Integration der bidirektionalen Schnittstelle ist es wichtig zu wissen, welche Informationen vom mechanischem Design Modell im System Modell benötigt werden und umgekehrt. Beide Modelle entwickeln sich über den Entwurfszyklus hinweg separat voneinander, sollen aber durch den Austausch in der Lage sein sich gegenseitig zu aktualisieren. Ändert sich beispielsweise das Aussehen eines Bauteils entweder im System oder im mechanischen Modell, muss das jeweilig andere Modell diese Information ebenfalls übernehmen. Des Weiteren kann das mechanische Modell im Laufe des Entwurfs Parameter wie Schwerpunkt, Trägheitsmomente oder Materialparameter an das System Modell weiterleiten. Dort spielen beispielsweise auch Bedingungen, wie Flächenkontakt, Kongruenz oder Offset, die zwischen den Bauteilen des Satelliten gesetzt werden können eine Rolle. Diese Informationen sind vorerst nicht für das System

Modell relevant und werden nur als neue Positionen der Bauteile an dieses übergeben. Je detaillierter das CAD Modell wird, desto mehr Konstruktionsbauteile, wie beispielsweise Schrauben, Gelenkverbindungen und Bohrungen lassen sich im Modell wiederfinden. Auch diese Informationen müssen nicht zwangsläufig an das System Modell übergeben werden, wenn sie auf dieses keinen besonderen Einfluss haben.

Auch im System Modell werden Informationen hinterlegt, welche für das mechanische Design vorerst uninteressant sind, wie zum Beispiel der Energieverbrauch einzelner Bauteile, der durch das Energie Subsystem im System Modell angelegt werden kann. Nur Informationen zum Aussehen, sowie zur Lage der Bauteile in der Gesamtkonfiguration des Satelliten sind für das mechanische Modell von Bedeutung.

Dennoch stellt das System Modell, die zentrale Verwaltung aller Informationen dar, da dort alle Domänen ihre spezifischen Daten hinterlegen. Aus diesem Grund soll es in Bezug auf den bidirektionalen Austausch nur vom System Modell aus möglich sein Daten zu löschen oder hinzuzufügen. Das mechanische Modell ist nur eine weitere Schnittstelle des System Modells. Durch den ständigen Austausch zwischen System Modell, mechanischem Modell und dem immer detailreicheren Austauschfluss wird es anhand der Kopplung von mechanischem und modellbasiertem Design möglich sein das System Modell für alle Domänen konsistent zu halten.

Informationsfluss Kriterien

Die Implementierung der Schnittstelle erfordert spezielle Anforderungen bezüglich der Informationsaustauschrichtung. Von bedeutender Rolle ist welche Informationen vom mechanischen Design Modell an das System Modell übergeben werden sollen und welche Informationen zurück vom System Modell an das mechanische Modell gehen. In dieser Arbeit wurden die sogenannten CRUD Kriterien genutzt um die Informationsflussrichtung zu spezifizieren. CRUD steht für: *Create*, *Replace*, *Update*, *Delete*. Die Abkürzung kommt aus dem Bereich der Datenbanken und bezieht sich auf die dort zu findenden Operatoren.[50] Die einzelnen Operatoren sind wie folgt zu verstehen:*Create* legt einen neuen Datensatz, *Replace* liest bzw. ersetzt einen bereits vorhandenen Datensatz und fügt in diesem Informationen an oder löscht diese, *Update* aktualisiert den bereits vorhandenen Datensatz mit einem aktuelleren Datensatz und *Delete* löscht diesen. Abbildung 5.3 zeigt den Einsatz der CRUD Kriterien für den Informationsfluss zwischen System Modell und mechanischem Modell.

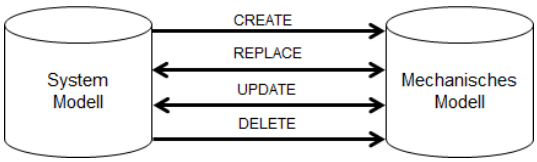

Abbildung 5.3: Die Integration unter Berücksichtigung von CRUD Kriterien

Wie in der Abbildung zu erkennen, soll es jedoch nicht von beiden Seiten der bidirektionalen Schnittstelle aus möglich sein alle Datenbankoperationen zu nutzen. Während das System Modell alle Datenbankoperatoren nutzen kann, ist das mechanische Modell nur in der Lage das System Modell zu aktualisieren

(Update) bzw. Informationen aus diesem zu ersetzen (Replace). Da das System Modell das zentrale Modell darstellt, welches alle Domänen Informationen bündelt, ist allein dieses in der Lage Gebrauch von den CRUD Kriterien Create und Delete zu machen. Dort können Bauteile des Satelliten aufbauend auf den gesammelten System Informationen neu angelegt oder auch entfernt werden. Beispielsweise kann der Lageregelungsingenieur die Information, dass er einen Sternensensor benötigt, an das System Modell weitergeben, nicht jedoch an das mechanische Modell. Nur im System Modell kann der neue Sternensensor definiert werden und über die Schnittstelle können die Informationen über den neu erstellten Sensor ebenfalls ins CAD Modell fließen.

Das aus dem System Modell erstellte CAD Modell, kann jedoch Änderungen vornehmen anhand der CRUD Kriterien Replace und Delete. Das mechanische Modell kann den, im System Modell definierten, Sternensensor geometrisch verändern, seine Lage in der Satellitenkonfiguration ändern oder ihm mechanische Designparameter, wie Material oder Masse geben.

Beispielsweise kann der mechanische Ingenieur nicht darüber entscheiden, einen weiteren Temperatursensor in das CAD Modell einzufügen. Diese Information muss vom Thermalingenieur kommen und dieser legt seine Informationen im System Modell ab. Es soll möglich sein aus dem System Modell ein mechanisches Design hervorzurufen, aus dem mechanischen Design soll allerdings kein System Modell erstellt werden, dieses soll lediglich dadurch aktualisiert werden können. Diese Folgerungen beziehen sich auf die im Abschnitt 5.3 diskutierten Annahmen zum Informationsfluss. Ein Beispiel für das Replace Kriterium wäre die Änderung des Sensors von der geometrischen Form einer Box in einen Zylinder. Eine Änderung der äußeren Gestalt des Zylinders, beispielsweise des Radius, fällt unter das Update Kriterium. Die Änderungsinformationen an dem Sternensensor können über die Schnittstelle zurück in das System Modell gegeben werden. Auch dort können die genannten Änderungen am Sensor erfolgen und wieder zurück in das mechanische Modell fließen, da auch das System Modell die Kriterien Replace und Update nutzen kann.

Kapitel 6

Implementierung der bidirektionalen Schnittstelle

In diesem Kapitel wird auf die Umsetzung der bidirektionalen Integration der Schnittstelle eingegangen, in dem zuerst die Architektur der Integrationsschnittstelle vorgestellt werden. Anschließend wird die Integration des mechanischen Designs durch die CAD Software CATIA in die MBSE Software Virtual Satellite unter Berücksichtigung der vorgestellten CRUD Kriterien dargestellt. Dabei wird sowohl die Implementierung aus CATIA Sicht, als auch aus Virtual Satellite Sicht vorgestellt.

6.1 Architektur der Integrationsschnittstelle

Abbildung 6.1 zeigt die Gesamtarchitektur der bidirektionalen Integrationsschnittstelle. Sie zeigt die beiden Werkzeuge der Schnittstelle, CATIA und Virtual Satellite. Während Virtual Satellite das System Modell erstellt, erstellt CATIA das mechanische Modell. In beiden Modellen soll es möglich sein, eine Visualisierung des Satelliten zu erzeugen. Dabei soll über die Schnittstelle ein bidirektionaler Austausch beider Modelle über ein geeignetes Austauschformat stattfinden.

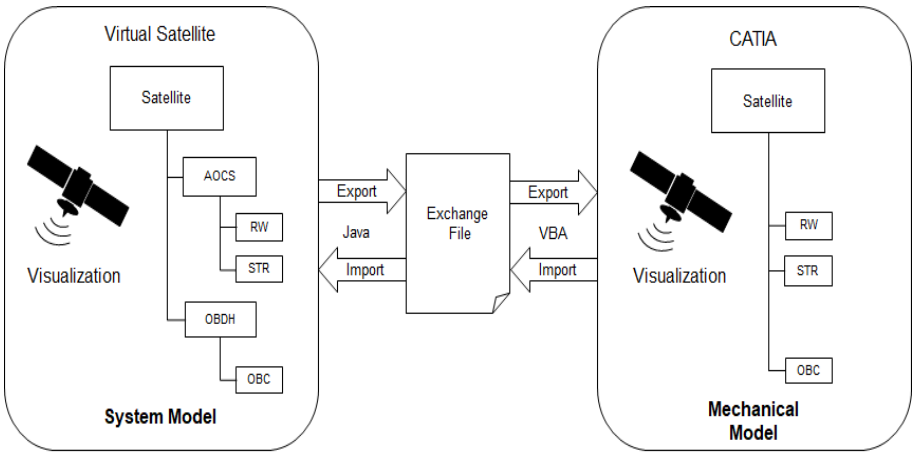

Abbildung 6.1: Architektur der bidirektionalen Schnittstelle

Während Virtual Satellite in Java entwickelt wird, bietet CATIA die Möglichkeit die Skriptsprache Visual Basic for Application (VBA) zu nutzen, so dass für das Austauschformat nicht nur native CAD Formate in Frage kommen. Um die Kopplung zwischen mechanischen Modell und System Modell zu realisieren, ist es wichtig zu wissen, welche Informationen, in diesem Fall, von Virtual Satellite in CATIA benötigt werden und umgekehrt. Abbildung 6.2 zeigt die Nutzung der Virtual Satellite Produktstrukturen in Verbindung mit CATIA.

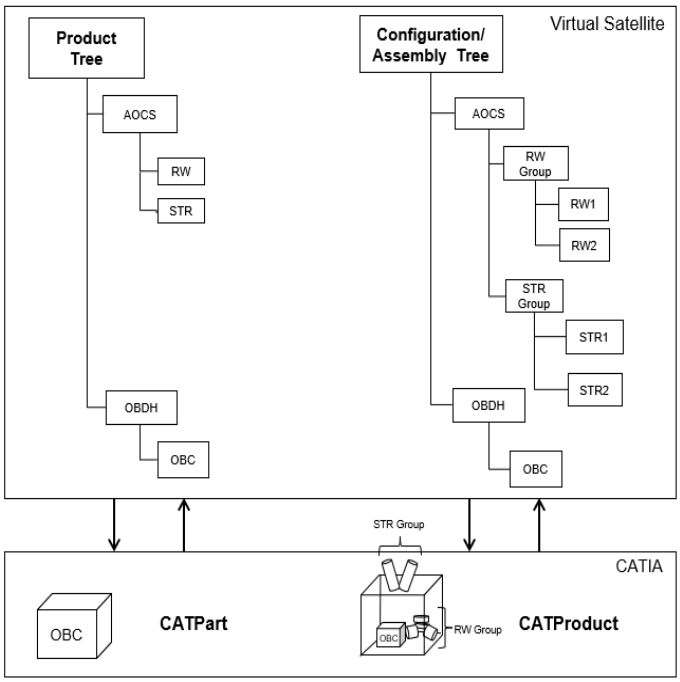

Abbildung 6.2: Austausch zwischen Virtual Satellite und CATIA

Im oberen Bereich sind die Produktstrukturen von Virtual Satellite zu erkennen, während im unteren Bereich die Produktstrukturen von CATIA zu erkennen sind. In Virtual Satellite werden im Product Tree alle Bauteile hinterlegt, welche im Satelliten integriert werden sollen, während der Configuration und der Assembly Tree eine Konfiguration dieser Bauteile beinhalten.(vgl. Kapitel 3) Eine ähnliche Zuordnung kann auch in CATIA vorgenommen werden. Während die CATParts die Bauteile des Satelliten darstellen, sind in den CATProducts die Positionen der CATParts im CATIA Assembly hinterlegt. Die Informationen aus dem Product Tree des System Modells fließen dabei in die CATParts von CATIA, während die Informationen aus Assembly bzw. Configuration Tree in die CATProducts von CATIA eingehen. Aus CATPart und CATProduct kann dann letztendlich die Satellitenkonfiguration als CATA Assembly erstellt werden, die dieselbe Konfiguration des Satelliten bereitstellt, wie der Assembly bzw. Configuration Tree in Virtual Satellite. Änderungen bezüglich des Aussehens eines Bauteils fließen in den Product Tree, während Änderungen zur Lage in der Satellitenkonfiguration in den Configuration bzw. Assembly Tree eingehen. Dabei ist zu beachten, dass durch die Vererbungsmechanismen (vgl. Kapitel 3) in Virtual

Satellite eine Änderung im Configuration Tree beispielsweise eine Positionsänderung von einem Bauteil in der Satellitenkonfiguration, ebenfalls eine Änderung der Bauteilposition im erbenden Assembly Tree zur Folge hat. Für die Implementierung dieser Schnittstelle wird die Informationsfluss-Richtung und die Austauschmethode zwischen CATIA und Virtual Satellite in den folgenden Kapiteln genauer betrachtet werden.

6.1.1 Umsetzung der CRUD Kriterien

Während in Virtual Satellite die genutzten Bauteile des Satelliten definiert werden, können diese an CATIA weitergegeben werden um dort Änderungen vorzunehmen. Beispielsweise ist es in CATIA möglich das Bauteil geometrisch zu verändern, so dass es sich besser in die mechanische Konfiguration des Satelliten integrieren lässt. Diese Information kann CATIA durch die bidirektionale Schnittstelle an das System Modell zurückgeben. Dennoch spielen nicht alle Informationen aus dem System Modell im mechanischen Modell eine Rolle und umgekehrt. Dabei werden für die Analyse des Informationsfluss die CRUD Kriterien, die im vorherigen Kapitel 5 vorgestellt worden sind, wieder aufgegriffen. Das nachfolgende Schaubild 6.3 zeigt eine Erweiterung der Abbildung 5.3 und erörtert die Funktion der Schnittstelle zwischen CATIA und Virtual Satellite unter Berücksichtigung der CRUD Kriterien.

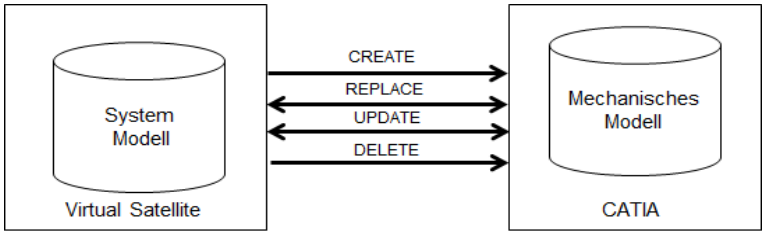

Abbildung 6.3: Austausch zwischen Virtual Satellite und CATIA unter Berücksichtigung der CRUD Kriterien

Wie in der Abbildung zu erkennen und in Kapitel 5 diskutiert, soll es nicht von beiden Seiten der Schnittstelle aus möglich sein alle Datenbankoperationen zu nutzen. Während Virtual Satellite alle vier Kriterien implementiert, implementiert CATIA nur die Kriterien Replace und Update. Das bedeutet aus Sicht von Virtual Satellite ist es möglich neue Bauteile im Satelliten anhand des Create Kriteriums zu erstellen. Diese Bauteile können dann auch ersetzt oder verändert werden mit Hilfe der Kriterien Replace und Update. Auch das Entfernen von Bauteilen aus Virtual Satellite Sicht ist durch das Delete Kriterium anwendbar. CATIA jedoch ist nur in der Lage durch die Kriterien Replace und Update Änderungen an der bestehenden Satellitenkonfiguration vorzunehmen. Die Kriterien Create und Delete sind aus Sicht von CATIA nicht implementiert worden, um die Anforderungen aus Kapitel 5.3 einzuhalten. Die Anforderungen beziehen sich darauf, dass der mechanische Ingenieur keinerlei Mitspracherecht in den Änderungen anderer Domänen bekommt, sondern diese Änderungen immer erst über das System Modell an das mechanische Modell, in diesem Fall das CAD Modell in CATIA, weitergegeben werden können.

Der erste Austausch zwischen CATIA und Virtual Satellite erfolgt als Export von den System Modell Informationen in das CAD Modell. Dabei erstellt CATIA das mechanische Design unter Berücksichtigung der System Modell Daten. Erst dann ist CATIA in der Lage die Kriterien Replace und Update zu nutzen und das System Modell zu aktualisieren bzw. bestimmte Komponenten zu ersetzen.

6.1.2 Austauschformat

Da nun geklärt ist unter welchen Kriterien der Informationsfluss zwischen Virtual Satellite und CATIA stattfindet, wird ein geeignetes Format benötigt in welchem die Informationen zwischen Virtual Satellite und CATIA über die Schnittstelle ausgetauscht werden können. Die im vorherigen Kapitel 4 vorgestellten Austauschformate und Methoden werden im Folgenden miteinander verglichen, um anschließend aus all diesen das geeignetste Verfahren bzw. das geeignetste Format zu wählen. Die nachstehende Tabelle zeigt noch einmal die wichtigsten Vor- und Nachteile der einzelnen Methoden und Formate auf.

Austauschformat/-Methode	**Vorteile**	**Nachteile**
RPC	leichte Implementierung	komplex abhängig vom Betriebssystem instabil
STEP	ISO-Standard	komplex hohe Datenmengen
STL	verschiedene Speicherformate	nur Transport von Basisinformationen
XML	sprachunabhängig komplexe Strukturen darstellbar weit verbreitet	hoher Speicherbedarf beschränkte Datentypisierung
JSON	sprachunabhängig komplexe Strukturen darstellbar erfolgreich in VSD implementiert	nicht validierbar weniger weit verbreitet als XML

Anhand eines Features zum Excel Export wurde die RPC Methode in der Vorgängerversion Virtual Satellite 4, Virtual Satellite 3, bereits getestet und führte zu keinem zufriedenen stellenden Ergebnis da sich gezeigt hat, dass die Implementierung einer Excel ActiveX Schnittstelle schwer zu steuern und instabil ist. Auch CATIA ist in der Lage die ActiveX Schnittstelle zu nutzen. Aus den obengenannten Gründen wurde die Methode des RPC in dieser Arbeit für die Integration der Schnittstelle jedoch nicht weiter analysiert.

Die beiden nativen CAD Formate stellen zwar gängige Formate beim Austausch von CAD-Daten dar. Das STEP Format ist beispielsweise eines der beliebteren Formate, wenn es um den Austausch in CAx-Systemen geht, da dieses über den reinen Geometrieaustausch hinausgeht und wesentliche Produktdaten darstellen und übertragen kann. Jedoch ist das STEP-Format aufgrund aller enthaltenen Produktdaten relativ komplex und nicht allzu leicht interpretierbar, zudem ist es eher für den Austausch zwischen CAD Programmen vorgesehen. Ein entscheidender Nachteil des STL-Austauschformats liegt darin, dass das Format nur Basisinformationen transportiert. Daten zu innerer Struktur, Farbe und Textur gehen verloren. Um die Oberfläche eines 3D Körpers mit dem STL Format so exakt wie möglich darstellen zu können werden viele kleine Dreiecksfacetten benötigt. Das führt zu einer sehr hohen Datenmenge. Handelt es sich um gekrümmte Körper, wie beispielsweise Kugeln oder Zylinder, bietet STL nur eine Annäherung der Oberfläche. Rein aus Visualisierungszwecken werden STL-Formate jedoch gerne genutzt, wie beispielsweise in der Virtual Satellite Visualisierung. Durch STL-Dateien ist es möglich nicht nur primitive geometrische Formen darzustellen, sondern auch abstraktere Formen zu visualisieren. Dabei sind die Basisinformationen des Formats vollkommen ausreichend. Die nativen CAD Formate lassen sich zwar von CATIA Seite einfach erzeugen, für die Schnittstellenintegration aus Sicht der MBSE Software muss diese jedoch angepasst werden, da die direkten Austauschformate eher für den Austausch zwischen CAD-Programmen vorgesehen sind und keine gängigen Austauschformate in anderen Programmen darstellen. Aus diesen Gründen wurde sich in der vorliegenden Arbeit für das json-Format entschieden, da es sich im Vergleich mit den anderen Austauschformaten hervorheben konnte. Mit dem Format lassen sich aufgrund der dort genutzten Datenstrukturen komplexe hierarchische Baumstrukturen in einfacher Weise darstellen. Da Virtual Satellite und auch CATIA hierarchische Baumstrukturen nutzen um das System des Raumfahrzeuges darzustellen, bietet sich das json-Format hier an. Zwar bietet auch das XML-Format die Möglichkeit komplexe Datenstrukturen darzustellen, der Vergleich von XML und json zeigt jedoch, dass das json-Format deutlich schlanker als das XML-Format ist und somit einen geringeren Speicherbedarf aufweist und im Wesentlichen auch deutlich schneller zu verarbeiten ist. Dazu kommt, dass das json-Format bereits erfolgreich in eine unidirektionale Kopplung zwischen MBSE und mechanischem Design, in der Software Virtual Spacecraft Design implementiert wurde.

Das nachfolgende Schaubild 6.4 demonstriert die Nutzung des json-Austauschformats zwischen der MBSE Software Virtual Satellite und der CAD Software CATIA. Da das json-Format von allen modernen Programmiersprachen unterstützt wird, bestehen sowohl für CATIA über die Visual Basics Applikation, als auch für den java-basierten Virtual Satellite die Möglichkeit einen Konverter bzw. eine Bibliothek in die Software einzubinden. Somit sind beide Programme in der Lage json-Dateien zu erstellen und exportieren, sowie einzulesen und importieren. Mithilfe der im vorherigen Kapitel in Abschnitt 4.2.5.3 vorgestellten json-Datenstrukturen wurde für diese Arbeit die benötigte json-Datei in zwei Teile unterteilt, das json-Objekt *json Products* und das json-Array *json Parts*. Welche Informationen in welche Datenstruktur fließen und wie diese Datenstrukturen aufgebaut sind beschreibt Abbildung 6.4.

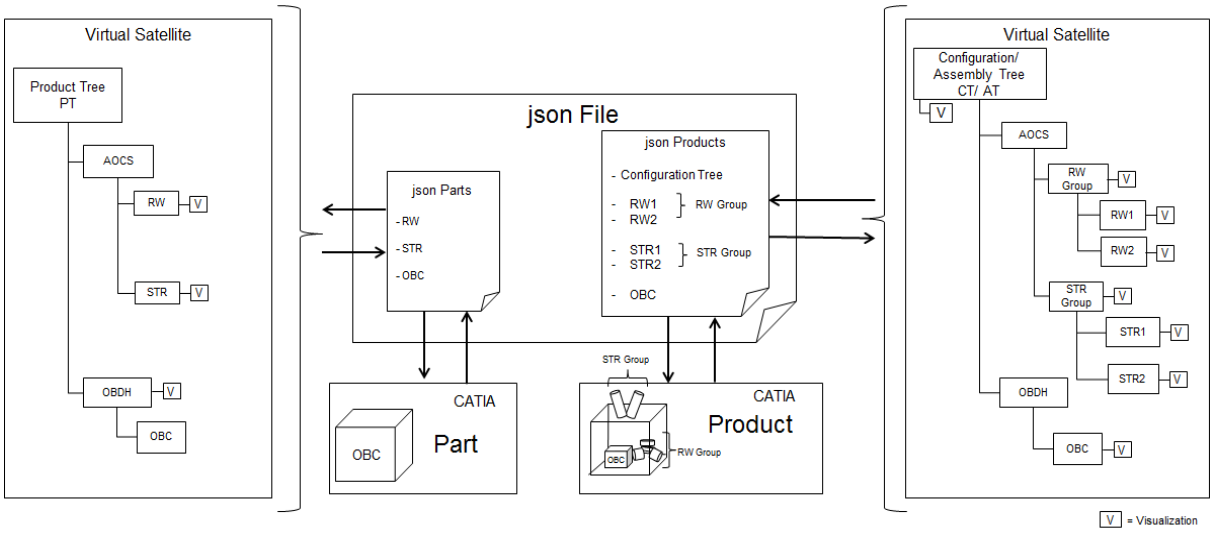

Abbildung 6.4: Das json-Format als Austauschformat zwischen Virtual Satellite und CATIA

Die Verbindung von Product Tree und CATPart wird über das json-Array „Parts“ realisiert, während das json-Objekt „Products“ den Austausch zwischen Assembly (Configuration) Tree und CATProduct übernimmt. Dabei befinden sich im json-Array alle hinterlegten Bauteile des Satelliten aus dem Product Tree und im json-Objekt alle Positionen dieser Bauteile in der Satellitenkonfiguration aus dem Assembly bzw. Configuration Tree.

Ein simples Beispiel soll den Nutzen des Austauschformats verdeutlichen. (vgl. Abbildung 6.5 und Abbildung 6.6) Zu sehen ist einfacher Satellit erstellt in Virtual Satellite. Im Product Tree befinden sich lediglich zwei Bauteile, ein Solarpanele und Reaktionsrad. Beide Bauteile werden im Configuration Tree wieder verwendet und in einer ersten Konfiguration integriert. Des Weiteren ist die Visualisierung des Configuration Trees in Virtual Satellite abgebildet. Das Reaktionsrad ist in der Abbildung als lila Kugel erkennbar, während die Solarpanele blaue Boxen darstellen.

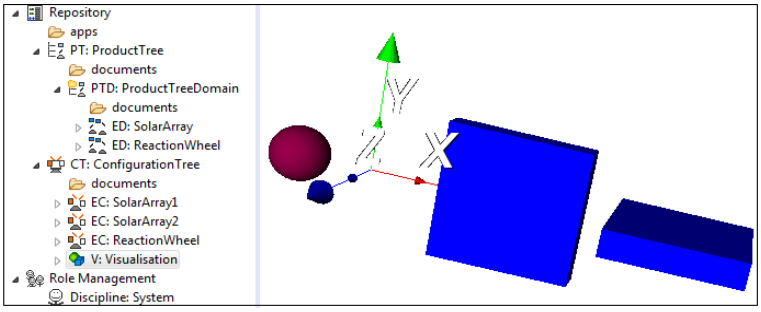

Abbildung 6.5: Beispiel einer simplen Satelliten Konfiguration in Virtual Satellite

Die zugehörige json-Datei in Abbildung 6.6) beinhaltet dabei die Informationen aus dem Product Tree im json-Array Parts und die Informationen aus dem Configuration Tree im json-Objekt Products. Das Reaktionsrad hinterlegt im json-Array beispielsweise nur Information zu dessen äußerer Erscheinung (Form, Farbe, Größe), während im json-Objekt die Lage des Reaktionsrads Position und Rotation hinterlegt ist.


```
"Parts":[
    {
        "shape":"SPHERE",
        "color":8388672,
        "name":"ReactionWheel",
        "lengthY":0.0,
        "lengthX":0.0,
        "radius":2.0,
        "uuid":"ec90535b-b758-4a6e-85cc-4be3f47b26ed",
        "lengthZ":0.0
    },
    {
        "shape":"BOX",
        "color":255,
        "name":"SolarArray",
        "lengthY":10.0,
        "lengthX":10.0,
        "radius":0.0,
        "uuid":"ab7d0b82-94d0-40be-a1fd-de6487ecb538",
        "lengthZ":2.0
    }
]
```

```
"Products":{
    "posX":0.0,
    "posY":0.0,
    "posZ":0.0,
    "rotX":0.0,
    "children":[
        {
            "posX":10.0,
            "posY":0.0,
            "posZ":0.0,
            "rotX":0.0,
            "rotZ":0.0,
            "rotY":0.0,
            "name":"SolarArray1",
            "uuid":"cb9ed889-cc2e-4c14-a575-cba34464ce94",
            "partName":"SolarArray"
        },
        {
            "posX":0.0,
            "posY":0.0,
            "posZ":0.0,
            "rotX":90.0,
            "rotZ":0.0,
            "rotY":0.0,
            "name":"SolarArray2",
            "uuid":"30baf7d4-b3e9-4c70-965a-8ec4c94eadac",
            "partName":"SolarArray"
        },
        {
            "posX":-5.0,
            "posY":0.0,
            "posZ":0.0,
            "uuidED":"ec90535b-b758-4a6e-85cc-4be3f47b26ed",
            "shape":"SPHERE",
            "rotX":0.0,
            "rotZ":0.0,
            "rotY":0.0,
            "name":"ReactionWheel",
            "uuid":"5f30705c-6cc5-4c50-b446-7e45d60cd511",
            "partName":"ReactionWheel"
        }
    ],
    "rotZ":0.0,
    "rotY":0.0,
    "name":"ConfigurationTree",
    "uuid":"33fe7fde-5ae8-49a3-aba1-603acdfa0ef4"
}
```


Abbildung 6.6: json-Format einer simplen Satellitenkonfiguration

Des Weiteren ist im json-Objekt Products der Eintrag „partName“ zu erkennen. Dieser json-Schlüssel gibt Aufschluss darüber von welchem Bauteil im Product Tree das aktuelle Bauteil der Konfiguration erbt. Diese Vererbung führt auf die Vererbungsarchitektur in Virtual Satellite zurück, in der beispielsweise der Configuration Tree die Bauteile des Product Trees erbt. Auch in CATIA wird diese Vererbungsarchitektur genutzt, in dem die CATProducts von den CATParts erben.

6.1.3 Modelltransformation

Der folgende Abschnitt beschäftigt sich mit der Implementierung einer Modelltransformation zur Realisierung der bidirektionalen Schnittstelle zwischen der CAD Software CATIA und der MBSE Software Virtual Satellite. Die implementierte Transformation basiert dabei auf einem zweischichtigen *Mapping*.

Die Mapping Methode ist dabei an die vorgestellte Modelltransformation in Kapitel 4.2.4 angelehnt. Beide Schritte des Mappings werden im Folgenden genauer vorgestellt.

Konzept Mapping

Der erste Schritt des Mappings ist das Konzept Mapping. Abbildung 6.7 zeigt das Prinzip des Konzept Mappings. Das Konzept Mapping bezieht sich auf die Visualisierung von Virtual Satellite und die Visualisierung von CATIA. Da durch das Visualization Konzept in Virtual Satellite bestimmte Parameter zum Aussehen einer Satellitenkomponente angelegt werden können, müssen diese auch in CATIA wieder auftauchen, um zu garantieren, dass beide Programme, dieselbe Visualisierung zugrunde legen. In CATIA wird das mit dem Parameterset realisiert. Ein Parameterset in CATIA ist ein Satz von Parametern, die für jedes CATPart angelegt werden können. Dabei sind die Parameter frei wählbar und können mit Formeln verknüpft werden, um eine Parametrisierung des CATParts vorzunehmen. Durch die Nutzung des json-Austauschformates können nun den angelegten Parametern, die vorhergesehenen Werte der Parameter mit übergeben werden, damit beide Programme dieselben Visualisierungsparameter aufweisen.

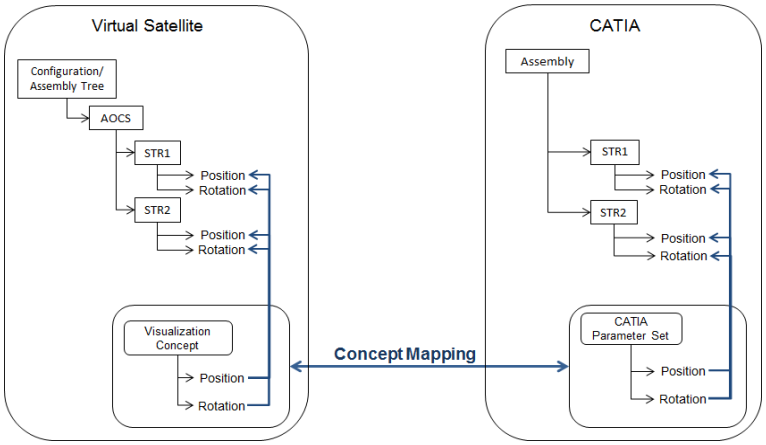

Abbildung 6.7: Theorie Konzept Mapping

Das nachfolgende Schaubild 6.8 zeigt auf der linken Seite das CATIA Parameter Set für das CATPart des Reaktionsrades (RW) und auf der rechten Seite die Visualisierungsparameter von Virtual Satellite für die dazugehörige Element Definition des Reaktionsrades. Das Parameterset und das Visualization Konzept weisen dieselben Parameter auf. Um die Äquivalenz der beiden Konzepte zu verdeutlichen wurden einige Parameter in rot hervorgehoben.

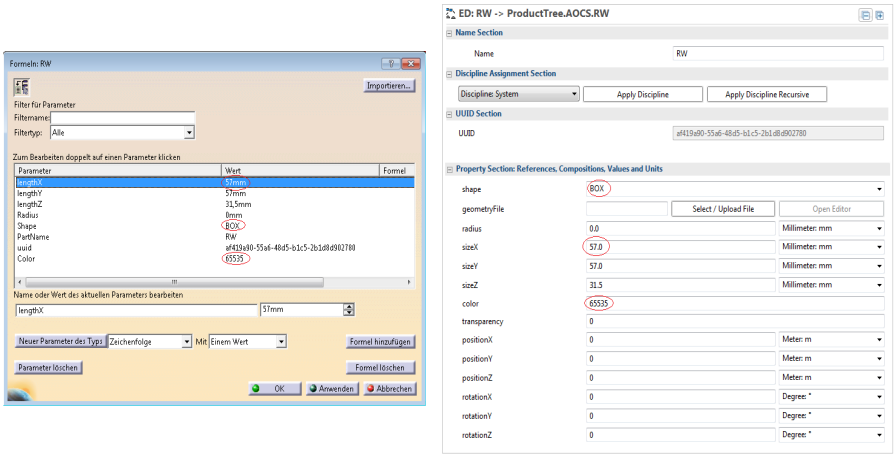

Abbildung 6.8: Anwendung Konzept Mapping

Dabei wird auf Product Tree Ebene das Konzept der Visualisierung für Form, Farbe und Größe genutzt. Während auf Configuration und Assembly Tree Ebene das Konzept der Visualisierung für Rotation und Position genutzt wird. Das Konzept Mapping ist dabei für alle Instanzen in der Satellitenkonfiguration identisch.

Instanz Mapping

Um das Konzept Mapping nun auf Instanz Ebene anwenden bzw. die Visualisierungsparameter eines bestimmten Bauteils in der Satellitenkonfiguration von Virtual Satellite an dasselbe Bauteil im CATIA Assembly übergeben zu können, wird der zweite Schritt der Modelltransformation benötigt, das Instanz Mapping. Abbildung 6.9 zeigt die theoretische Anwendung des Instanz Mappings.

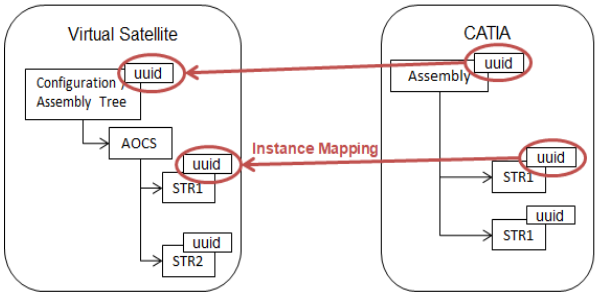

Abbildung 6.9: Theorie Instanz Mapping

Um eine Zuordnung der Komponenten und Konfigurationen zwischen CATIA und Virtual Satellite zu gewährleisten ist es wichtig, diese eindeutig zu kennzeichnen. In Virtual Satellite wird dafür für jede Komponente und jeden Product, Configuration und Assembly Tree eine sogenannte *UUID (Universally*

Unique Identifier) erstellt. Dort wird niemals dieselbe UUID zweimal angelegt, jede UUID ist universal eindeutig. Die UUID wird von Virtual Satellite im System Modell angelegt und kann nicht geändert werden, CATIA muss diesen Wert als Parameter hinterlegen, damit in jeder weiteren Entwurfsphase des Roundtrip Engineerings immer wieder auf diese bestimmte Komponente zugegriffen werden kann. Mit Hilfe dieser UUID ist es möglich einzelne Instanzen aus der gesamten Struktur direkt zu identifizieren. Wird beispielsweise ein Sensor als Kugel im Virtual Satellite angelegt und in CATIA exportiert, gibt Virtual Satellite über den Export auch die zugehörige UUID in der json-Datei mit. Die Informationen aus der json-Datei speichert CATIA im Parameterset ab, darunter auch den Parameter der UUID. Ändert sich nun im Virtual Satellite das Bauteil und der Sensor soll einen Zylinder darstellen, wird auch die json-Datei beim Reexport an die Änderungen angepasst, die UUID bleibt jedoch dieselbe. Durch die hinterlegte UUID kann CATIA den Sensor aufsuchen und identifizieren und die anderen Parameter im Parameterset durch das Konzept Mapping anpassen. So wird aus der Kugel ein Zylinder. In der Abbildung 6.10 ist das Instanz Mapping an derselben Abbildung wie im oberen Abschnitt gezeigt, nun aber auf Instanz Ebene bezogen. Die UUID der gezeigten Instanz des Reaktionsrades ist sowohl im Visualisierungskonzept von Virtual Satellite, als auch im Parameterset von CATIA grün hervorgehoben.

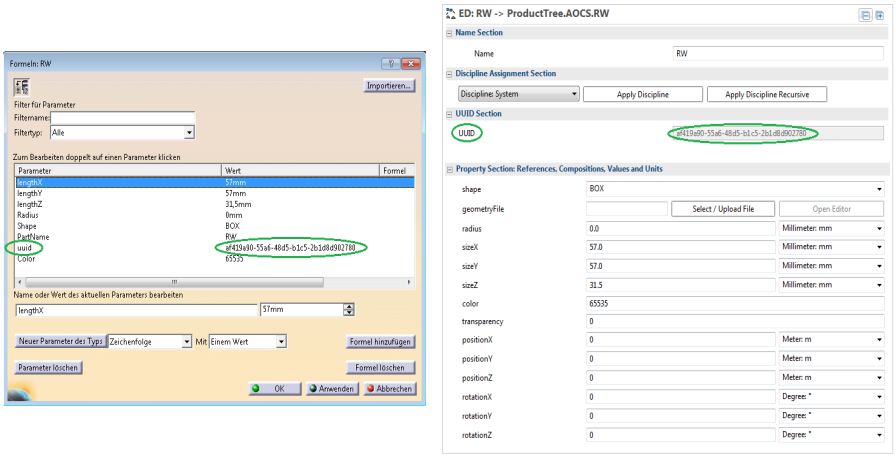

Abbildung 6.10: Instanz Mapping

Das Mapping zwischen Virtual Satellite und CATIA funktioniert nur in dem beide Schritte des Mappings ausgeführt werden. Nur das Instanz bzw. das Konzept Mapping reichen nicht aus um eine vollständige Modelltransformation zwischen mechanischem Design und System Modell herzustellen.

6.2 Implementierung in Virtual Satellite

Für diese Arbeit wurde das Virtual Satellite 4 Research Konzept genutzt, welches auf dem Core-Konzept der Software basiert und Spielraum für Experimente und wissenschaftliche Erweiterungen lässt ohne

in die produktive Software einzugreifen. (vgl. Kapitel 3.2) Um das json-Austauschformat zu nutzen wurde die *json-simple library* für Java mit in den Code eingebunden. Um den Export und Import von Assembly bzw. Configuration Tree in Virtual Satellite via json-Format zu gewährleisten wurden in Virtual Satellite Research zwei Superklassen für den Import und Export erstellt. Abbildung 6.11 zeigt das UML Aktivitätsdiagramm für den Export. Das UML Diagramm zeigt hier nur die Algorithmen für den Export eines Configuration Trees.

Um den Export aus Virtual Satellite zu ermöglichen greift der Algorithmus als erstes auf die Methode *getAllElementDefinitionsFromTree* zu, um alle Bauteile des Satelliten im Configuration Tree, welche vom Product Tree erben zu sammeln und im json-Array Parts zu hinterlegen. Berücksichtigt werden dabei nur Bauteile, die ein angehängtes Visualisierungskonzept besitzen, das wird durch die Methoden *jsonPartsFromED* und *jsonPartsFromVis* realisiert. Im json-Array werden dann die Parameter aus dem Visualisierungskonzept bezüglich der äußeren Erscheinung des Bauteils hinterlegt. Dazu zählen Form, Farbe und Größe. Des Weiteren wird für jedes Bauteil zusätzlich die UUID mit im json-Format hinterlegt, welche für das Instanz Mapping benötigt wird. Im zweiten Schritt werden alle Bauteile aus dem Configuration Tree im json-Objekt Products mit der Methode *getAllElementConfigurations* hinterlegt. Zu den hinterlegten Parametern gehört hier jetzt nicht mehr die äußerliche Erscheinung des Bauteils, sondern dessen Lage in der Satellitenkonfiguration. Dazu zählen unter anderem Position und Rotation. Des Weiteren wird die Information abgelegt von welchem Bauteil im Product Tree das aktuelle Bauteil der Konfiguration erbt. Auch hier wird wieder zusätzlich die UUID jeder Komponente im Configuration Tree übergeben, um das Konzept Mapping auf Instanz Ebene auszuführen. Zusätzlich zu allen hinterlegten Komponenten im Configuration Tree, wird ebenfalls der Configuration Tree selbst mit Position und Rotationsdaten im json-Objekt Products hinterlegt. Diese Information wird in CATIA benötigt um die Erstellung eines Ursprungs für das Assembly zu generieren. Alle Bauteile werden dann relativ zu diesem Ursprung in das Assembly integriert. Da es sich bei den genutzten Produktstrukturen um verschachtelte Baumstrukturen handelt, sind die beiden Methoden rekursiv geschrieben, so dass der komplette Configuration Tree mit all seinen Kindern für das json-Format berücksichtigt werden kann. Wenn alle Informationen aus Product und Configuration Tree in den zwei json Datenstrukturen hinterlegt sind, wird aus dem json-Objekt Products und dem json-Array Parts mit der Methode *transfromCT* eine json-Datei erstellt.

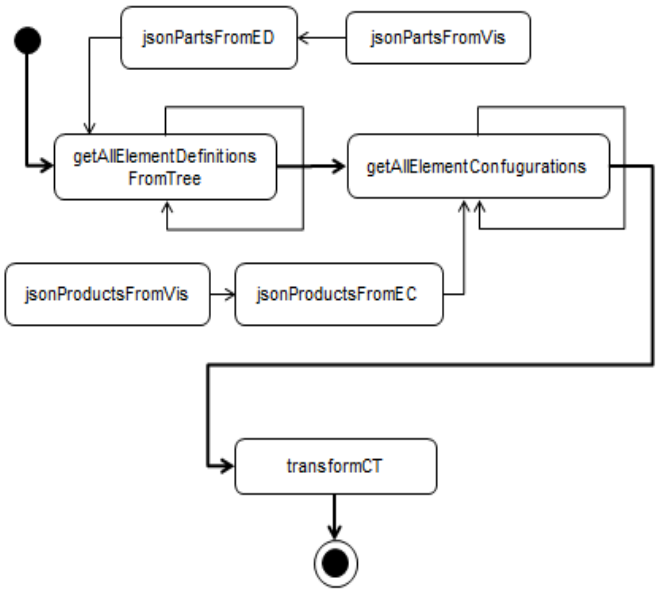

Abbildung 6.11: Export eines Configuration Trees

Abbildung 6.12 zeigt das UML Diagramm für den Import auf Configuration bzw. Assembly Tree. Für den Import des CAD Modells in Virtual Satellite muss zuerst das json-Format gelesen werden, dies erfolgt mit der Methode *readJsonFile*. Aus der json-Datei werden dann die beiden Datenstrukturen des json-Objekts und json-Arrays herausgegriffen. Damit werden dann im späteren Verlauf aus dem json-Array Parts die Informationen der Eigenschaften der verwendeten Bauteile des Satelliten mit deren äußerlicher Form ausgelesen und an den Product Tree anhand der Methode *getEDFromJson* übergeben. Das sind die Element Definitions in Virtual Satellite. Im json-Objekt Products werden die Informationen der Eigenschaften zur Lage der Bauteile in der Gesamtkonfiguration mit Hilfe der Methode *getECFromJson* bzw. *getECFromJson* ausgelesen. Die Informationen werden in Virtual Satellite an die Element Configurations oder Element Occurences weitergegeben, je nachdem in welchen Tree importiert wurde. Um diese Eigenschaften aus dem CAD Modell den richtigen Bauteilen im System Modell zuzuordnen, wird in der json-Datei zu jeder Komponente (Element Definition, Element Configuration, Element Occurence) ebenfalls dessen UUID hinterlegt. Diese wird in den Methoden *getMatchingED* und *getMatchingEC* bzw. *getMatchingEO* verwendet um das CAD Bauteil dem passenden System Modell Bauteil zuzuordnen. Anschließend wird die komplette json-Datei in den jeweiligen Tree importiert mit der Methode *transformJsonFile*.

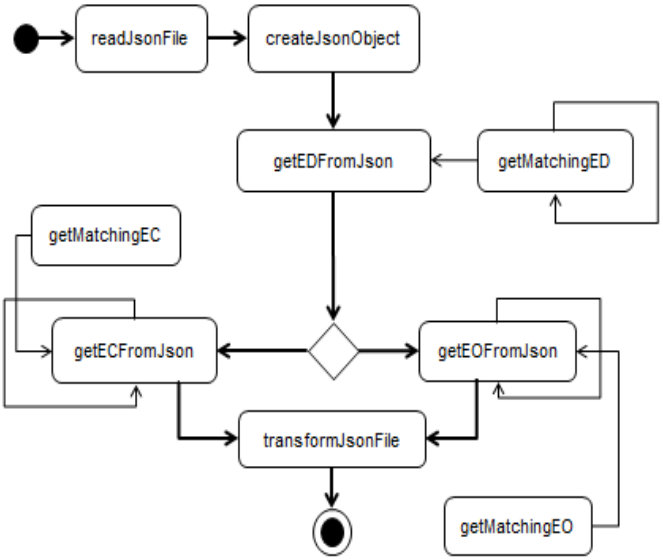

Abbildung 6.12: Import von Configuration bzw. Assembly Tree

Die Behandlung von Configuration und Assembly Tree im Export bzw. Import erfolgt weitestgehend identisch. Ein einziger Unterschied besteht im Zugriff auf die Bauteile des Satelliten, welche im Product Tree als Element Definitions hinterlegt werden. Während der Configuration Tree direkt vom Product Tree erbt und so direkt auf die Element Definitions im Product Tree zugreifen kann, muss der Assembly Tree über den Configuration Tree auf den Product Tree und so auf die dort hinterlegten Element Definitions zugreifen. Die nachstehende Abbildung 6.13 verdeutlicht den Vererbungsprozess unter den Produktstrukturen von Virtual Satellite anhand der Thermalkontrollsystem Komponente eines Radiators.

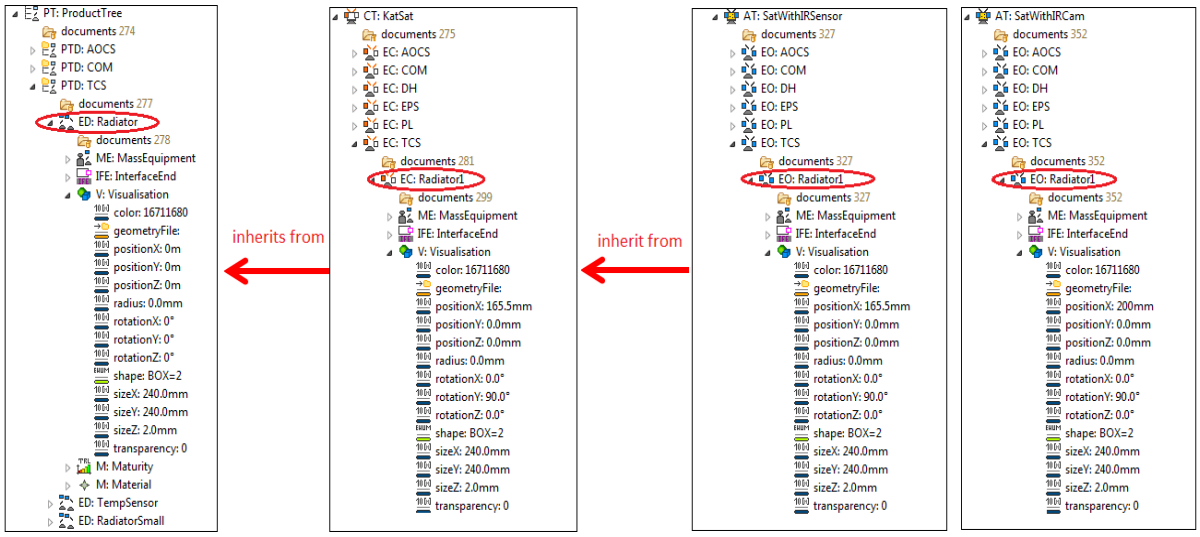

Abbildung 6.13: Vererbungshierarchie der Produktstrukturen

Dies erfolgt im Algorithmus anhand der Methode*getMatchingED*, die über den Configuration Tree die Product Tree Informationen sammelt. Dabei holt sich die aktuelle Element Occurence, die das Bauteil im Assembly Tree repräsentiert, die Element Configuration aus dem Configuration Tree aus welchem der Assembly Tree erbt. Die Element Configuration gibt dann wiederum Aufschluss darüber von welcher Element Definition diese im Product Tree erbt. Da der Assembly Tree vom Configuration Tree erbt bedeutet ein Import der CAD Modell Informationen in den Configuration Tree, in dem beispielsweise Bauteile im Assembly verschoben worden sind, gleichzeitig eine Änderung für den Assembly Tree und dessen Satellitenkonfiguration. Dabei ist der Algorithmus so ausgelegt, dass beim Import auf einen Assembly Tree Positionsänderungen nur in diesem stattfinden. Beim Import auf einen Configuration Tree werden Positionsänderungen sowohl im Configuration als auch im Assembly Tree vorgenommen. Anders verhält es sich wenn nicht die Position der Bauteile, sondern deren Darstellungsinformationen sich durch das mechanische Design verändern. Diese Information geht sowohl beim Import auf den Assembly Tree, als auch beim Import auf den Configuration Tree in den Product Tree. Da bei beiden Importmethoden zuerst die Methode *getMatchingED* aufgerufen wird, welche sich die im Product Tree hinterlegten Element Definitions holt, die die äußere Gestalt der Satellitenbauteile beschreiben. Sobald die Information einer Element Definition im Product Tree geändert wird, wird die Information direkt an Assembly und Configuration Tree weitergeleitet und das Bauteil erhält in beiden Konfigurationen die neue geänderte Gestalt. Ein Beispiel für den Import auf die verschiedenen Tree Konzepte wird in Kapitel 7 gegeben.

JUnit Tests

Für Import und Export wurden mittels JUnit Softwaretests erstellt. Tests sind kleine Programme, die ohne Benutzerkontrolle automatisch über die Quellcodebasis laufen und anhand von Regeln zeigen, dass gewünschte Teile sich so verhalten wie gewünscht. Für die Test Cases wurde eine kleine Studie angelegt, in der sich ein Product Tree mit ein paar Bauteilen und ein Configuration Tree mit einer ersten Satellitenkonfiguration befinden. Durch die Test Cases soll gezeigt werden, dass sowohl der Import als auch der Export anhand des json-Austauschformats funktioniert. Dabei wurden die Test Cases nach und nach erweitert. Beispielsweise wurden die beiden erstellten Datenstrukturen, das json-Objekt Products und das json-Array Parts der json-Datei vorerst einzeln getestet, bevor die komplette json-Datei importiert bzw. exportiert wurde. Auch die Behandlung von Configuration und Assembly Tree wurde einzeln betrachtet. Abbildung 6.14 zeigt einen Testdurchlauf anhand der JUnit Tests. Zu sehen ist, dass der Testdurchlauf anscheinend nicht erfolgreich war. Das ist an der roten Statusleiste rechts oben zu erkennen. Im linken Bereich finden sich die Informationen dazu, welche Test Cases nicht erfolgreich waren, während der rechte Bereich darauf eingeht, was im jeweiligen Test Case schief gelaufen ist.

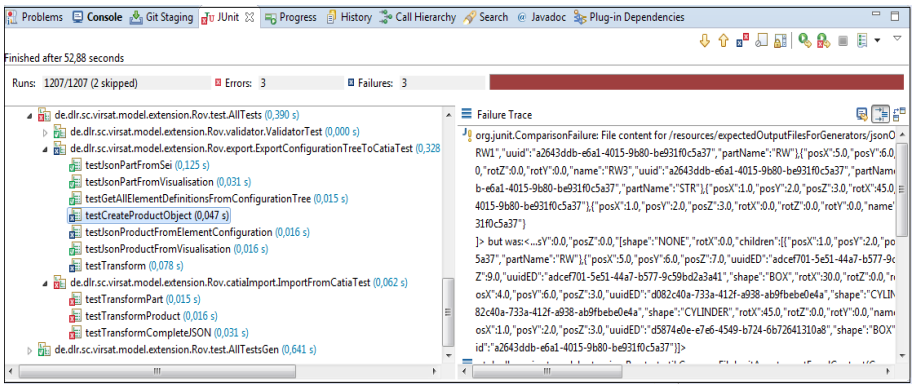

Abbildung 6.14: JUnit Testdurchlauf

6.3 Implementierung in CATIA

Um das json-Format in CATIA ein- und auszulesen wird die Skriptsprache VBA (Visual Basic for Applications) genutzt. Mit Hilfe dieser wurden Makros erstellt. Für den Import und Export von CATIA aus wurden mittels VBA zwei Hauptmakros erstellt, welche verschiedene Untermakros zur Realisierung des Lesens und Schreibens von json-Dateien enthalten. Das Makro *ReadJSON* und das Makro *WriteJSON*. Beide Makros wurden als Icons auf der CATIA Schaltfläche integriert, um dem Nutzer den Export und Import des System Modells zu erleichtern ohne direkt auf den VBA Code zugreifen zu müssen. Die Icons der Schaltfläche sind in Abbildung 6.15 zu erkennen.

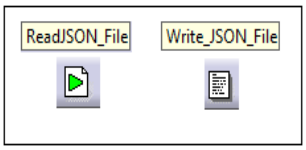

Abbildung 6.15: Hauptmakros als Icons an der CATIA Oberfläche

Um das json-Format nutzen zu können wurde im angelegten Visual Basic Projekt neben den Hauptmakros ein *json-Converter* für VBA integriert. In Abbildung 6.16 sind die beiden Hauptmakros als UML Aktivitätsdiagramme dargestellt. CATIA behandelt beim Export und Import Assembly Tree und Configuration Tree gleich. Beide Baumkonzepte können als CATIA Assembly dargestellt werden.

Beim Einlesen des json-Formats werden die Informationen aus dem json-Array Parts, an ein Untermakro zur Erstellung eines CATParts gegeben *CATPartCreater/Updater*. Dabei prüft das Makro, ob die CATParts bereits vorhanden sind, wenn dies der Fall ist wird das CATPart nicht neu erstellt, sondern das Vorhandene wird mit den Informationen aus der json-Datei aktualisiert. Dies ermöglicht einen Reimport vom System Modell in das CAD Modell. Dasselbe gilt für das Makro *ProductCreater/Updater*, welches sich aus der json-Datei das json-Objekt Products holt und die dort enthaltenen Informationen in ein CATProduct umwandelt. Aus den erstellten bzw. aktualisierten CATParts und CATProducts wird dann ein CATIA Assembly erstellt bzw. mit den Änderungen aus der json-Datei aktualisiert.

Zum Schreiben einer json-Datei wird der umgekehrte Weg genommen. Das Assembly wird zuerst Schritt für Schritt durchgegangen um dort alle enthaltenen CATParts und CATProducts in das json-Format zu transformieren. Dabei wird beim Erstellen des json-Arrays Parts auf die Parametersets der CATParts zugegriffen in denen sich die Visualisierungsparameter zur äußeren Erscheinung der Bauteile befinden, die in Virtual Satellite benötigt werden. Für das json-Objekt Products werden auf die Positionen der CATParts und CATProducts im CATIA Assembly zugegriffen. Positionen im Assembly lassen sich in CATIA nur in einer Transformationsmatrix ausgeben. Wie der Umgang mit der Transformationsmatrix in dieser bidirektionalen Schnittstelle anhand des json-Austauschformats gehandhabt wird, wird im folgenden Unterabschnitt erörtert.

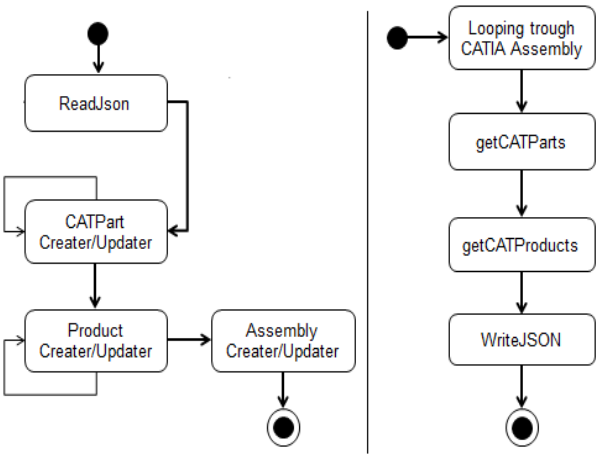

Abbildung 6.16: Hauptmakros ReadJSON & WriteJSON

Des Weiteren wurden in CATIA zusätzliche mechanische Parameter hinterlegt, welche durch CAD Tools leicht zu generieren sind, wie beispielsweise Material, Massenschwerpunkt und Massenträgheitsmomente. Diese Parameter werden ebenfalls mit in der json-Datei hinterlegt und können durch die Einbindung eines neuen Konzeptes in das System Modell übergeben werden. Das generierte Konzept wurde aufbauend auf den Prinzipien der Engineering Categories erstellt und in die Virtual Satellite Struktur implementiert. Dabei erhielt das Konzept den Namen *Catia Properties*. Abbildung 6.17 zeigt den Aufbau des *Catia Properties* Konzepts im Quellcode. Das Konzept ist momentan dafür ausgelegt Eigenschaften wie Massenschwerpunkte, sowie Material und Trägheitstensoren an die Bauteile bzw. Trees in Virtual Satellite anzuhängen, kann jedoch einfach erweitert werden.

```
Concept de.dlr.sc.virsat.model.extension.Rov displayname "CatiaProperties" version 2.0
    Import de.dlr.sc.virsat.model.extension.ps.*;

    // Get material of part selected in CATIA
    Category Material description "Material from CATIA"{
       Applicable For All;
       Cardinality 1;
       StringProperty material;
    }

    Category Matrix {
       FloatProperty xx default 0.0;
       FloatProperty xy default 0.0;
       FloatProperty xz default 0.0;
       FloatProperty yx default 0.0;
       FloatProperty yy default 0.0;
       FloatProperty yz default 0.0;
       FloatProperty zx default 0.0;
       FloatProperty zy default 0.0;
       FloatProperty zz default 0.0;
    }

    // Get Center of Gravity calculated from CATIA
    Category CenterOfGravity description "COG of SubAssemblies and AssemblyTree or ConfigurationTree"{

       Cardinality 1;
       FloatProperty centerOfGravityX quantityKind "Length" unit "Millimeter" default 0.0;
       FloatProperty centerOfGravityY quantityKind "Length" unit "Millimeter" default 0.0;
       FloatProperty centerOfGravityZ quantityKind "Length" unit "Millimeter" default 0.0;
    }

    // Get moment of inertia matrix 3x3 from CATIA
    Category MomentOfInertias {

       Applicable For All;
       Cardinality 1;
       Type moiMatrix of Category Matrix;
    }

}
```

Abbildung 6.17: Konzept CATIA Properties

Positionsänderung in CATIA - Die Transformationsmatrix

Bei Änderungen betreffend der Position bzw. Rotation einer Komponente in der Gesamtkonfiguration muss in CATIA mit Transformationsmatrizen gerechnet werden, das gilt sowohl für den Weg von CATIA nach Virtual Satellite als auch anders herum. Das liegt daran, dass CATIA Bauteile eines Assemblies nur anhand von Transformationsmatrizen automatisiert rotiert und positioniert werden können. Und die genaue Lage eines Bauteils im Assembly nur als Transformationsmatrix ausgegeben werden kann. Um die Rotationswinkel aus der Transformationsmatrix zu bekommen, wird die Matrix beim Export aus CATIA an die json-Datei weitergegeben, danach erfolgt im Virtual Satellite Code eine Umrechnung der Transformationsmatrix basierend auf den nachfolgenden Berechnungen.

Die Transformationsmatrix stellt dabei eine 3x4 Matrix dar, dessen erste drei Spalten die Rotationen um die drei Hauptachsen, in X-, Y- und Z-Richtung darstellen. Die letzte Spalte beinhaltet die Translation der Komponente in X-, Y- und Z-Richtung und kann direkt ausgelesen werden. Mit Hilfe der Rotationsmatrix ist es allerdings trotzdem möglich, die entsprechenden Rotationswinkel um jede Achse auszulesen.[45] Die unten dargestellte Matrix T stellt eine solche 3x4 Transformationsmatrix schematisch dar.

$$T = \begin{pmatrix} m_{00} & m_{01} & m_{02} & t_0 \\ m_{10} & m_{11} & m_{12} & t_1 \\ m_{20} & m_{21} & m_{22} & t_2 \end{pmatrix} \tag{6.1}$$

Die m-Einträge der Matrix stellen die Rotationsmatrix dar. Diese setzt sich folgendermaßen aus den drei aufeinanderfolgenden Rotationen zusammen: Rotation θ_1 um die X-Achse, Rotation θ_2 um die Y-Achse

und Rotation θ_3 um die Z-Achse :

$$R_x(\theta_1)R_y(\theta_3)R_z(\theta_3) = \begin{pmatrix} c_2c_3 & c_2s_3 & -s_2 \\ s_1s_2c_3 - c_1s_3 & s_1s_2s_3 + c_1c_3 & s_1c_2 \\ c_1s_2c_3 + s_1s_3 & c_1s_2s_3 - s_1c_3 & c_1c_2 \end{pmatrix} \tag{6.2}$$

mit $c_1 = cos(\theta_1)$, $s1 = sin(\theta_1)$, etc.

Um nun an die passenden Rotationswinkel zu gelangen sind verschiedene Fälle zu betrachten. Der erste Fall ist der allgemein Fall, aus dem die drei Rotationswinkel aus folgenden Gleichungen entnommen werden können. (vgl.(6.3))

$$\begin{aligned} \theta_1 &= atan2(m_{12}, m_{22}) \\ \theta_2 &= atan2(-m_{02}, c2) \text{ mit } c2 = \sqrt{m_{00}^2 + m_{01}^2} \\ \theta_3 &= atan2(m_{01}, m_{11}) \end{aligned} \tag{6.3}$$

Der allgemeine Fall führt zu Problemen, im Fall, dass m_{00} und m_{01} infinitesimal klein oder gar Null sind, da dies zur Folge hat, dass auch m_{12} und m_{22} sehr klein werden. Dadurch wird die Berechnung von θ_1 problematisch. Aus diesem Grund wird der berechnete Wert für c_2 gegen einen infinitesimal kleinen Schwellenwert getestet. Wenn dieser Schwellenwert unterschritten wird, reduzieren sich die Matrixelemente der Rotationsmatrix auf ungefähr das Folgende:

$$\begin{pmatrix} 0 & 0 & -(\pm 1) \\ \pm s_1 & c_1 & 0 \\ \pm c_1 & -s_1 & 0 \end{pmatrix}$$

Das hat zur Folge, dass θ_1 nun mit folgender Formel berechnet wird (vgl. (6.4)) um das sogenannte *Gimbal Lock Problem* zu umgehen:

$$\begin{aligned} \theta_1 &= atan2(-m_{21}, m_{11}) \\ \theta_2 &= atan2(-m_{02}, c2) \text{ mit } c2 = \sqrt{m_{00}^2 + m_{01}^2} \\ \theta_3 &= atan2(m_{01}, m_{11}) \end{aligned} \tag{6.4}$$

Handelt es sich bei der Rotation um eine reine Rotation um die Z-Achse, so treten folgende Formeln in

Kraft (vgl. (6.5)).

$$\begin{aligned} \theta_1 &= atan2(m_{12}, m_{22}) \\ \theta_2 &= atan2(-m_{02}, c2) \text{ mit } c2 = \sqrt{m_{00}^2 + m_{01}^2} \\ \theta_3 &= atan2(s_1 \cdot m_{20} - c_1 \cdot m_{10}, c_1 \cdot m_{11} - s_1 \cdot m_{21}) s_1 = sin(\theta_1),\ c_1 = cos(\theta_1) \end{aligned} \tag{6.5}$$

Somit lassen sich alle nötigen Rotationen um die Achsen mittels der Transformationsmatrix berechnen. Alle drei Fälle sind im Java Code von Virtual Satellite implementiert. Beim Import der in CATIA erzeugten json-Datei werden die Winkel aus der Virtual Satellite Visualisierung aus der Transformationsmatrix, welche in der json-Datei enthalten ist, berechnet.

Für den Export von Virtual Satellite in CATIA werden die Rotationswinkel der Virtual Satellite Visualisierung zusammen mit den Positionsdaten in der Visual Basic Application durch Makros zurück in die Transformationsmatrix gerechnet. Die berechnete Transformationsmatrix kann dann genutzt werden um die Bauteile innerhalb des CATIA Assemblies zu verschieben.

Ein weiteres Problem der Positionsänderung mittels Transformationsmatrizen liegt darin, dass Rundungsfehler auftauchen können. Durch die Berechnung der Rotationswinkel aus der Transformationsmatrix kann es beispielsweose passieren, dass ein Winkel von 10°einen tatsächlichen Wert von 9.9999°aufweist. Bei vielen Iterationen zwischen System Modell und mechanischem Modell kann es daher passieren, dass der Rundungsfehler sich laufend vergrößert. Um das Problem weitestgehend zu umgehen, wurden im Virtual Satellite Code Rundungsfunktionen implementiert, so dass Winkel nur ganzzahlig auftauchen können.

Kapitel 7

Erprobung und Anwendungsszenarien

Für die Analyse der bidirektionalen Anbindung von mechanischem Design an den übergeordneten modellbasierten Systementwurf wurde ein bestehender Testdatensatz eines Satelliten anhand der Software Virtual Satellite genutzt. Ein Satellit bzw. eine Raumfahrzeug lässt sich immer in mehrere Subsysteme gliedern, dabei gibt es für jedes Subsystem einen Experten. Die Testmission beinhaltet dabei alle für einen Satelliten relevanten Subsysteme. Die Mission weist ein Thermalsystem (TCS), ein Lageregelungssystem (AOCS), ein Energiesystem (EPS), ein Datenverarbeitungssystem (DH), eine Nutzlast (PL), ein Kommunikationssystem, ein Struktursystem und ein Verkabelungssystem (FEA) auf. Auf den Aufbau der Testmission anhand von Virtual Satellite wird im Folgenden näher eingegangen.

7.1 Aufbau der Testmission in Virtual Satellite

Im Testdatensatz befinden sich ein Product Tree, ein Configuration Tree, und zwei Assembly Trees. Im Product Tree befinden sich als Product Tree Domains die einzelnen Subsysteme des Satelliten. Diese sind auch im Configuration Tree als Element Configurations und in den Assembly Trees als Element Occurences wieder zu finden. (vgl. Abbildung 7.1)

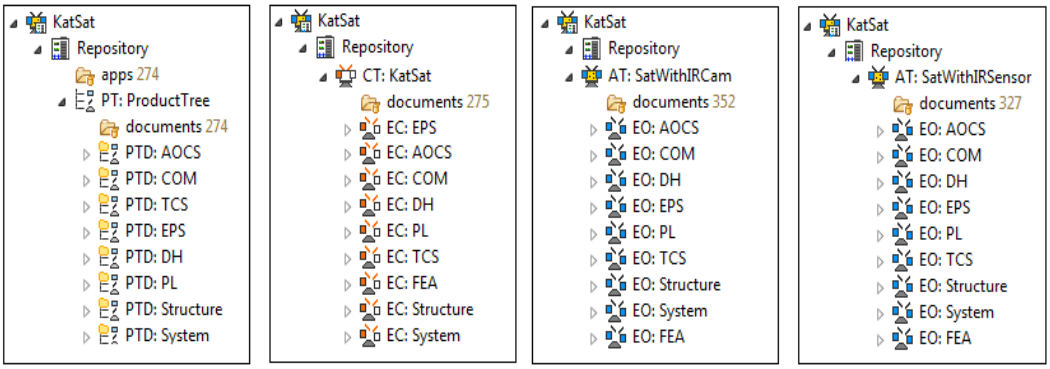

Abbildung 7.1: Aufbau Product Tree, Configuration Tree & Assembly Trees

Die beiden Assembly Trees sollen dabei eine Satellitenkonstellation bestehend aus zwei Satelliten darstellen. Während der erste Assembly Tree einen Infrarot Sensor als Nutzlast besitzt, besitzt der zweite Assembly

Tree eine Infrarot Kamera als Nutzlastkomponente. Die restlichen Subsysteme sind identisch aufgebaut und beinhalten dabei diverse Komponenten. Das Thermalkontrollsystem (TCS) beispielsweise besitzt unter anderem einen Radiator. (vgl. Abbildung 7.2) Dieser weist verschiedene Eigenschaften auf. Darunter auch Visualisierungseigenschaften zur Beschreibung der äußeren Erscheinung und der Lage des Radiators in der Satellitekonfiguration, die auf dem Konzept der *Visualization* (vgl. Kapitel 3) beruhen. Der Radiator als Bauteil wird im Product Tree definiert, so dass Configuration und Assembly Tree die Eigenschaften des Radiators erben.

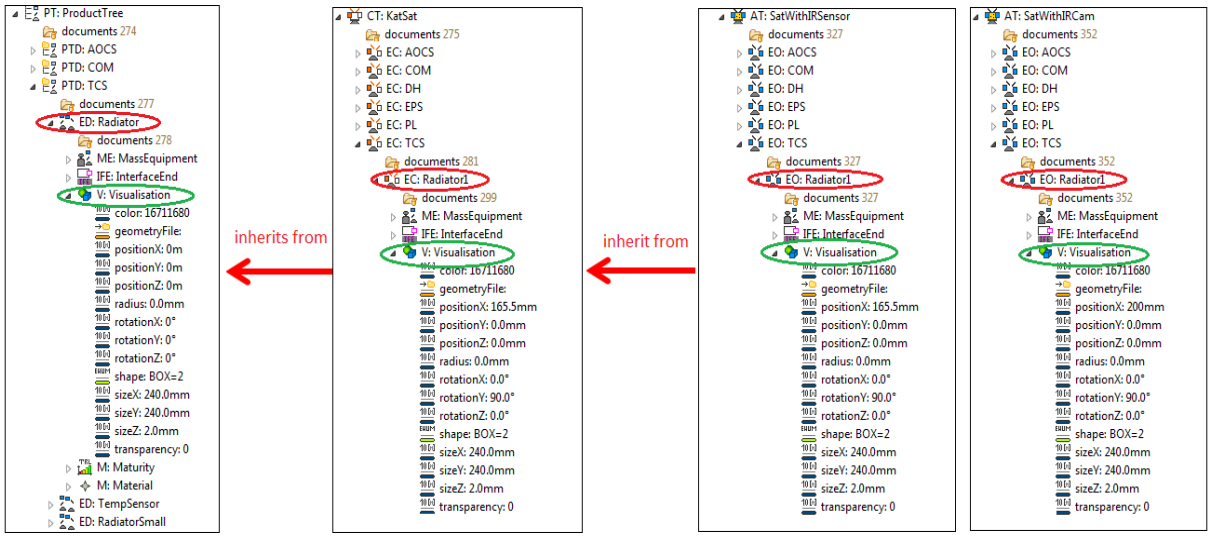

Abbildung 7.2: Radiator Bauteil in den einzelnen Trees

7.2 Informationsaustausch zwischen CATIA und Virtual Satellite

Anhand des Testdatensatzes wurden unterschiedliche Anwendungsszenarien der bidirektionalen Schnittstelle erprobt. Aufbauend auf der Diskussion zur Nutzung der CRUD Kriterien ist der erste Austausch des Roundtrip Engineering, der Export des System Modells in Virtual Satellite und dessen Import in CATIA, da das mechanische Design aus dem System Modell generiert werden soll. Beim erstmaligen Import der json-Datei, die in Virtual Satellite erzeugt wurde, wird das mechanische Design in CATIA als Assembly abgelegt. Bei jedem weiteren Austausch werden das mechanische Design, sowie das System Modell aktualisiert und an den neusten Stand angepasst, dabei können die Änderungen von beiden Seiten erfolgen. Nachfolgend werden einige Anwendungsfälle vorgestellt. Dabei werden sowohl Export, Import als auch Reexport und Reimport in beiden Modellen betrachtet.

7.2.1 Export und Reexport von Virtual Satellite nach CATIA

Der erste Austausch ist der Export des Virtual Satellite System Modells in CATIA. Um diesen Austausch zu ermöglichen wird das Create Kriterium beim Export verwendet. Durch die bidirektionale Schnittstelle ist es nicht nur möglich das System Modell in ein CAD Modell zu exportieren, es ist ebenfalls möglich das generierte CAD Modell bei einem Reexport zu aktualisieren. Beim Reexport können dann ebenfalls die anderen CRUD Kriterien: Replace, Update und Delete genutzt werden. Nachfolgend werden einige Anwendungsszenarien aufgezeigt.

Anlegen eines neuen Bauteils - Create

Das Hinzufügen von neuen Bauteilen im Satelliten ist nur in Virtual Satellite in dessen System Modell möglich. Abbildung 7.3 zeigt das Hinzufügen eines neuen Bauteils beispielhaft anhand eines Sternensensors. Dieser ist in Virtual Satellite als hellblauer Zylinder mit seinen zugehörigen Visualisierungsparametern erstellt worden. Des Weiteren ist die dazugehörige json-Datei erkennbar. Diese wird in Virtual Satellite exportiert und in CATIA eingelesen um die System Modell Informationen im CAD Modell darzustellen. Im json-Format stehen dabei alle benötigten Darstellungsinformationen des Sternensensors. Des Weiteren befindet sich in der json-Datei die UUID des Sternensensors, um diesen eindeutig zu identifizieren.

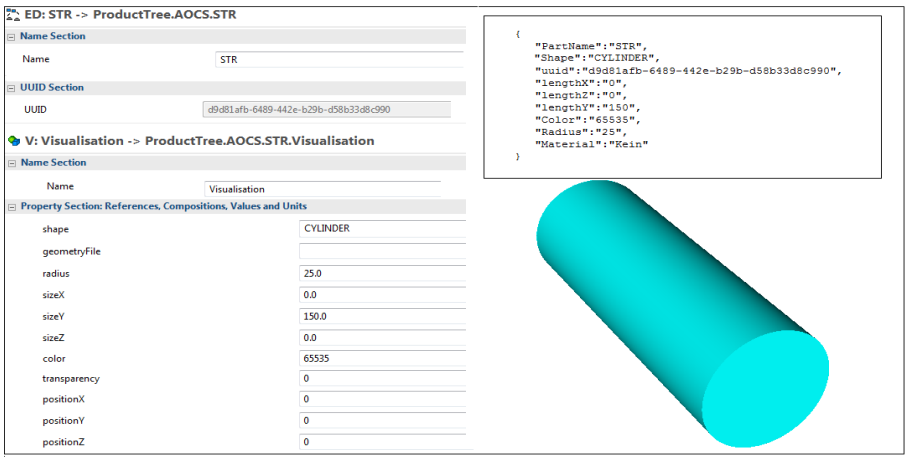

Abbildung 7.3: Create: Anlegen eines Sternensensors im System Modell

Abbildung 7.4 zeigt den Sternensensor als importiertes CATPart im CATIA Modell. Zu sehen ist auch das Parameterset des Sternensensors, das dieselben Parameterwerte enthält, wie die Visualisierung in Virtual Satellite. Diese wurden über die json-Datei in das CAD Modell importiert. Auch die Virtual Satellite UUID des Sternensensors wird im Parameterset hinterlegt.

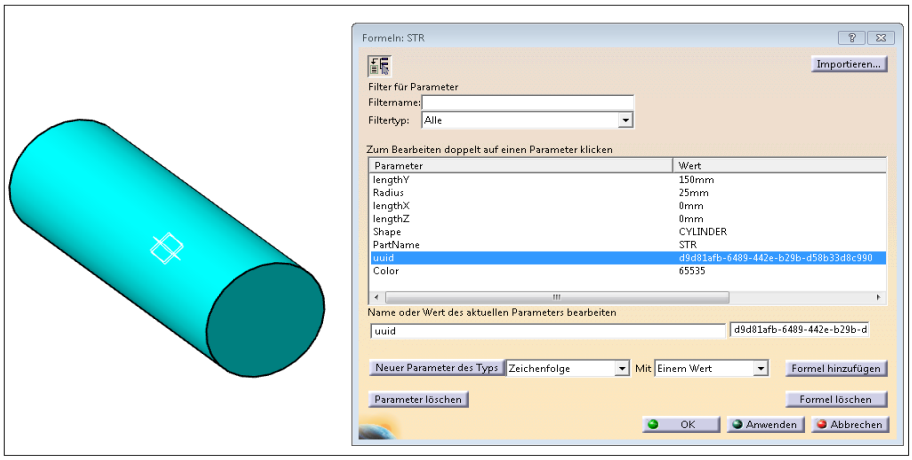

Abbildung 7.4: Exportierter Sternensensor im CAD Modell

Ersetzen und Ändern eines Bauteils - Replace & Update

Ein weiterer Anwendungsfall ist das Ersetzen und Ändern eines Bauteils. Diese beiden Anwendungsfälle fallen unter die CRUD Kriterien Replace und Update. Um auf das Beispiel des Sternensensors im vorherigen Abschnitt zurück zukommen, wurde dessen Zylinder Form in Virtual Satellite durch eine STL Datei ersetzt. Das STL Format wird in Virtual Satellite genutzt, um Bauteile über primitive geometrische Formen hinaus beschreiben zu können. (vgl. Kapitel 3) Des Weiteren wurde die Farbe des Sensors von hellblau auf grün geändert. (vgl. Abbildung 7.5) Zu sehen ist nicht nur die neu erstellte json-Datei, sondern auch der Ablageort der STL-Datei. Die STL-Datei, die in Virtual Satellite integriert wurde um den Sensor zu erstellen, wird dabei an einem Ablageort kopiert, den auch CATIA erreichen kann. In diesem Fall das H-Laufwerk. So sind beide Modelle in der Lage auf dieselbe STL-Datei zuzugreifen.

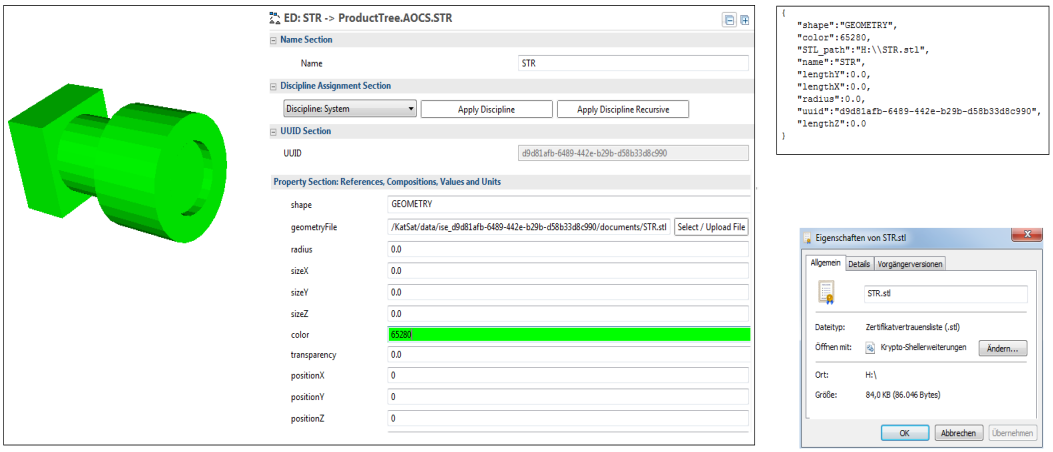

Abbildung 7.5: Replace: Ersetzen eines Bauteils von primitiver Form in STL Datei

Abbildung 7.6 zeigt den exportierten Sternensensor als STL Datei in CATIA. Dort sind im Parameterset des Bauteils die Parameter zu finden, die mit Hilfe der json-Datei vom System Modell in das CAD Modell übertragen worden sind.

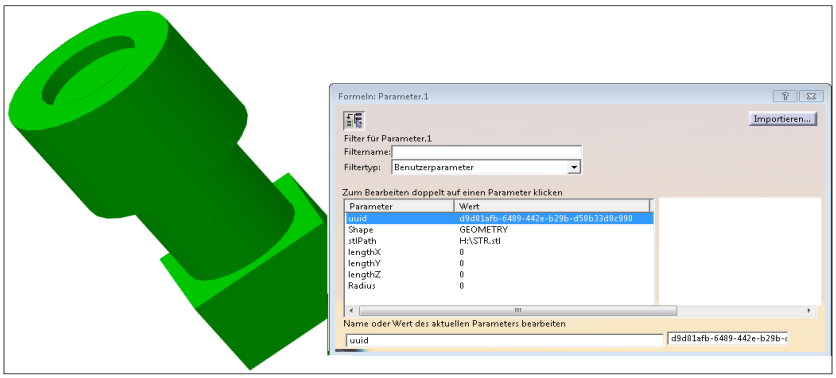

Abbildung 7.6: STL Datei in CAD Modell

Löschen eines Bauteils - Delete

Das letzte CRUD Kriterium Delete bezieht sich auf das Entfernen von Bauteilen aus dem Satelliten. Dieses ist nur von Virtual Satellite Seite aus anwendbar (vgl. Kapitel 5.3) und spielt daher nur beim Export bzw. Reexport in CATIA eine Rolle. Abbildung 7.7 zeigt den Löschvorgang eines Satellitenbauteils. Auf der linken oberen Seite der Abbildung ist die Visualisierung und der Aufbau eines Configuration Trees erkennbar. In diesem befindet sich im Subsystem des Thermalkontrollsystems (TCS) ein Radiator. Dieser ist in der Visualisierung auf der Frontseite des Satelliten als kleines rotes Rechteck dargestellt. Aus thermalspezifischen Gründen soll der Radiator aus der Satellitenkonfiguration entfernt werden. Der untere linke Teil der Abbildung zeigt den Satelliten mit dem entfernten Radiator. Dieser ist sowohl im Thermalkontrollsystem, als auch in der Visualisierung in Virtual Satellite nicht mehr dargestellt. Die rechte Seite der Abbildung zeigt den selben konfigurierten Satelliten in CATIA. Oben ist der alte Satellit erkennbar, der aus dem System Modell erstellt wurde als dieser noch einen Radiator besaß, während im unteren Bereich der Satellit nach dem Import des geänderten System Modells erkennbar ist. Auch in der Produktstruktur von CATIA ist der Radiator nicht mehr auffindbar. Durch das Entfernen des Radiators im System Modell wird beim Reexport eine neue json-Datei erstellt. In dieser befinden sich nun keine Informationen bezüglich des Radiators mehr.

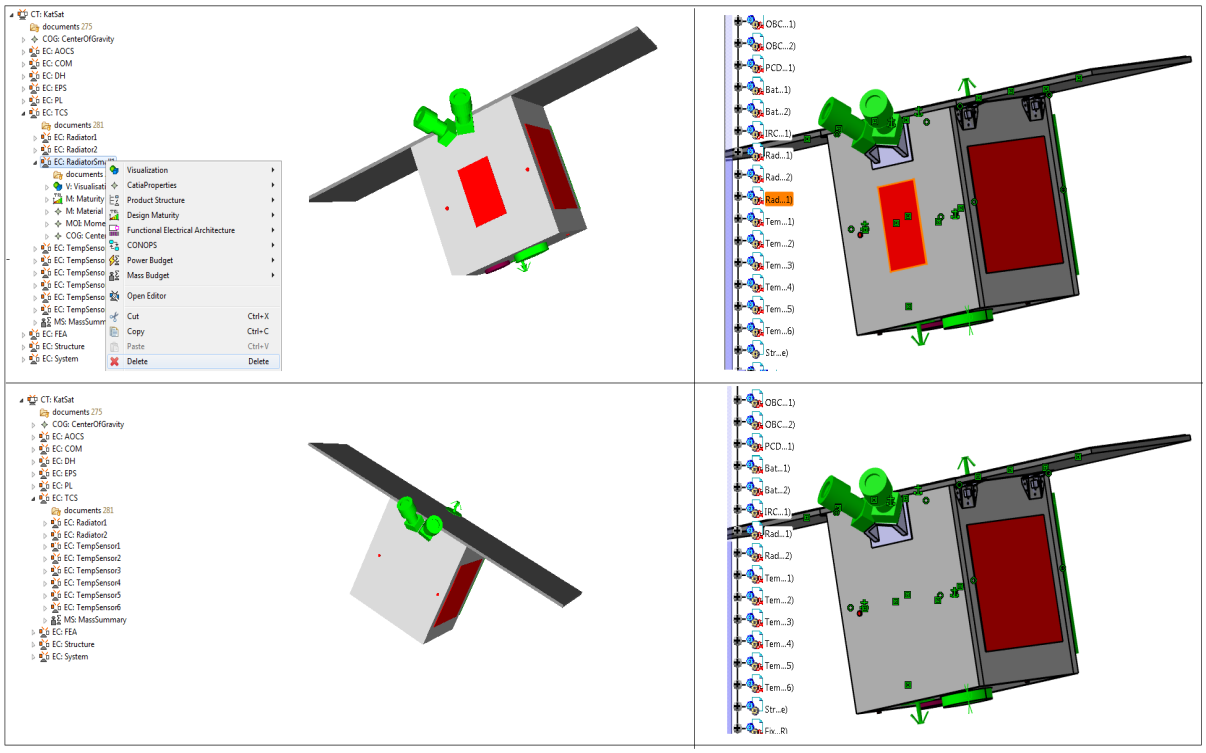

Abbildung 7.7: Delete: Entfernen eines Bauteils

Da sich die Bauteilinformationen nicht länger in der json-Austauschdatei befinden, kann der Radiator auch aus dem CAD Modell entfernt werden. Zu erkennen ist jedoch, dass der Satellit im CATIA Assembly beim Reexport des System Modells seine Konstruktionsbauteile und mechanischen Bedingungen beibehält, auch wenn am System Modell Änderungen vorgenommen worden sind. Diese Bauteile und Bedingungen sind jedoch nicht im System Modell wiederzuerkennen.

7.2.2 Import und Reimport von CATIA nach Virtual Satellite

Weitere Anwendungsszenarien der bidirektionalen Schnittstelle lassen sich anhand von Import und Reimport von CATIA nach Virtual Satellite darstellen. Im CAD Modell selbst werden dabei nur die CRUD Kriterien Replace und Update genutzt. Dabei erfolgt dieser Import erst nachdem erstmaligen Export von Virtual Satellite nach CATIA.

Import von Assembly & Configuration Tree

Beim Import in Virtual Satellite ist zu beachten, dass je nachdem in welchen Tree importiert wird, Änderungen auf die anderen Trees erfolgen. Beispielsweise hat der Import eines Configuration Trees

Auswirkungen auf den Assembly Tree, da dieser vom Configuration Tree erbt. Der Import auf einen Assembly Tree, welcher nur Positionsänderungen beinhaltet, hat jedoch keinerlei Auswirkungen auf den Configuration Tree. Lediglich bei Änderungen bezüglich des äußeren Erscheinungsbilds eines Bauteils werden die Änderungen beim Import auf einen Assembly Tree auf den Product Tree übertragen. Da die Bauteile im Configuration Tree vom Product Tree erben und das Bauteil sich im Product Tree geändert hat, wird auch das Bauteil im Configuration Tree geändert. Das nachstehende Beispiel verdeutlicht diesen Fall. Es wird ein Assembly Tree exportiert und in CATIA importiert. Dort werden Änderungen an der Satellitenkonfiguration und an den Bauteilen vorgenommen. Die neue Konfiguration wird dann mit Hilfe der json-Datei in den Assembly Tree reimportiert. Abbildung 7.8 zeigt den Satelliten in Virtual Satellite. Zu sehen sind der Configuration Tree und die beiden Assembly Trees vor dem Import der mechanischen Designinformationen aus dem CAD Modell. Hierbei steht das Thermalkontrollsystem (TCS) und dessen Komponente, ein Radiator, im Vordergrund. Die Konfigurationen sind identisch.

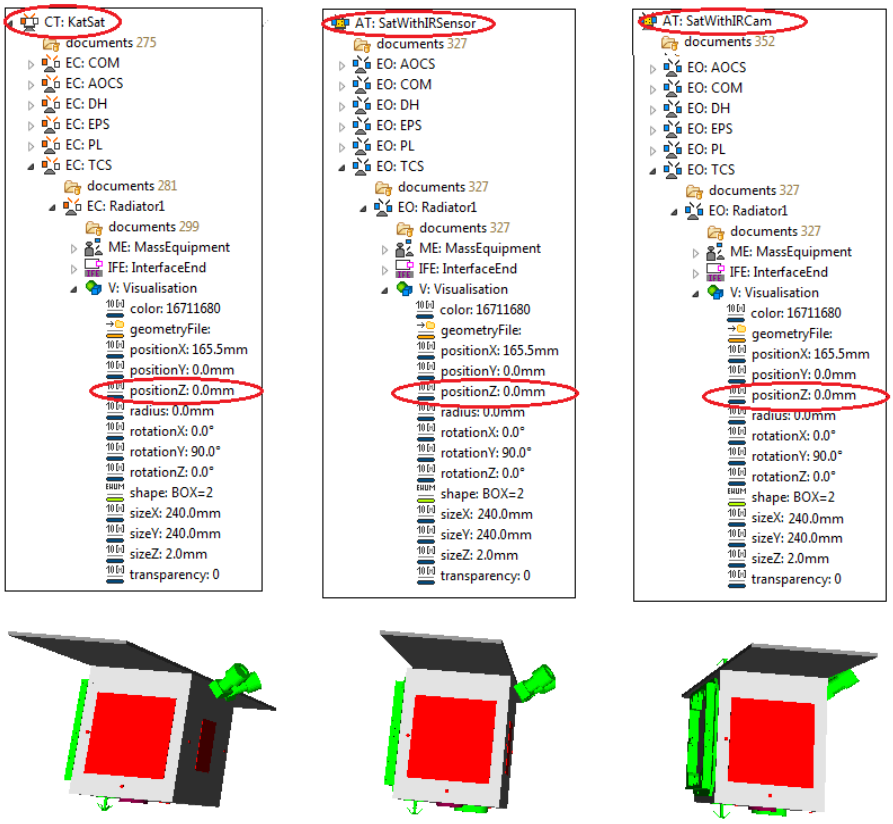

Abbildung 7.8: Trees vor Import

Abbildung 7.9 zeigt den Radiator noch einmal explizit als Komponente des Thermalkontrollsystems im Product Tree und dessen Visualisierungsparameter in Virtual Satellite. Unter der Visualisierung ist das zugehörige json-Format erkennbar, welches die Visualisierungsparameter des Radiators beinhaltet. Die In-

formationen aus dem System Modell werden nun exportiert. Dazu wird der Assembly Tree *SatWithIRCam* ausgewählt und mit Hilfe des json-Austauschformats exportiert.

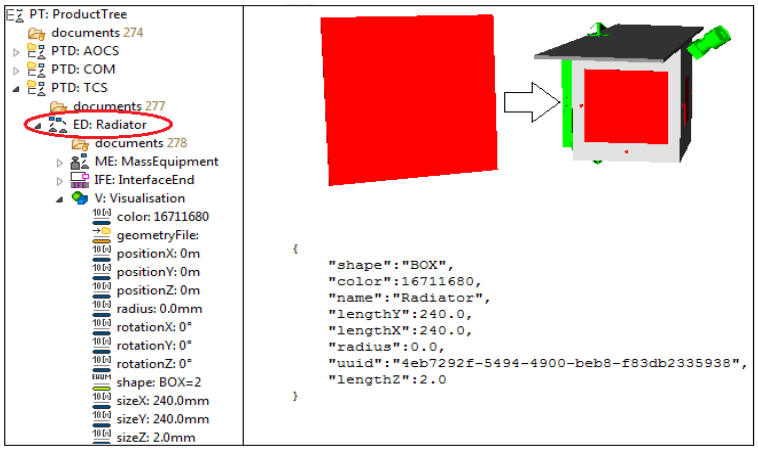

Abbildung 7.9: Radiator vor Import

Abbildung 7.10 zeigt die Satellitenkonfiguration als CAD Modell in CATIA und das dort enthaltene Parameterset des Radiators. Das CAD Modell wurde aus den System Modell Informationen anhand des exportierten json-Formats erzeugt und weist im Parameterset dieselben Informationen auf wie die Visualisierungsparameter in Virtual Satellite.

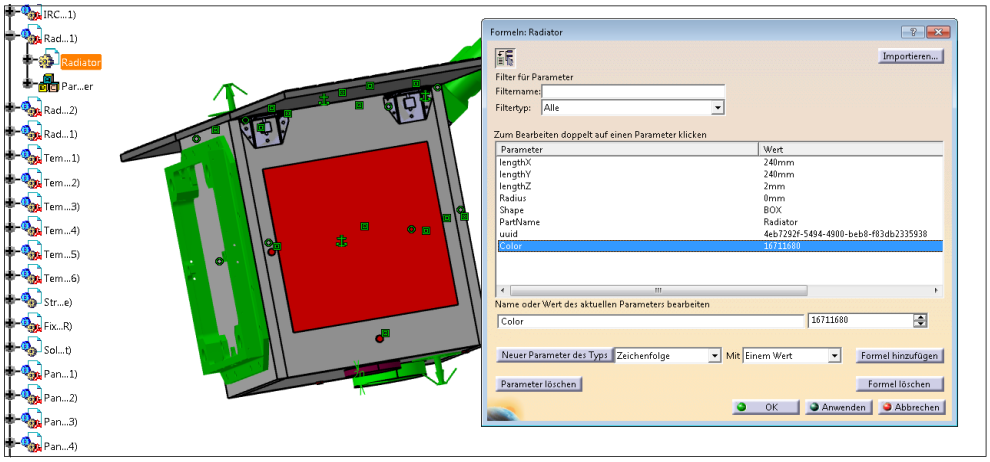

Abbildung 7.10: CAD Modell nach Export aus Virtual Satellite

Nun werden Änderungen an der Satellitenkonfiguration und insbesondere an der TCS Komponente des Radiators im CAD Modell vorgenommen. Das Bauteil wird in der Konfiguration verschoben und äußerlich verändert, aufbauend auf den beiden CRUD Kriterien Replace und Update. Abbildung 7.11 zeigt die Änderungen im CAD Modell. Die Informationen aus dem CAD Modell werden nun erneut in

eine json-Datei exportiert um diese anschließend in Virtual Satellite zu reimportieren.

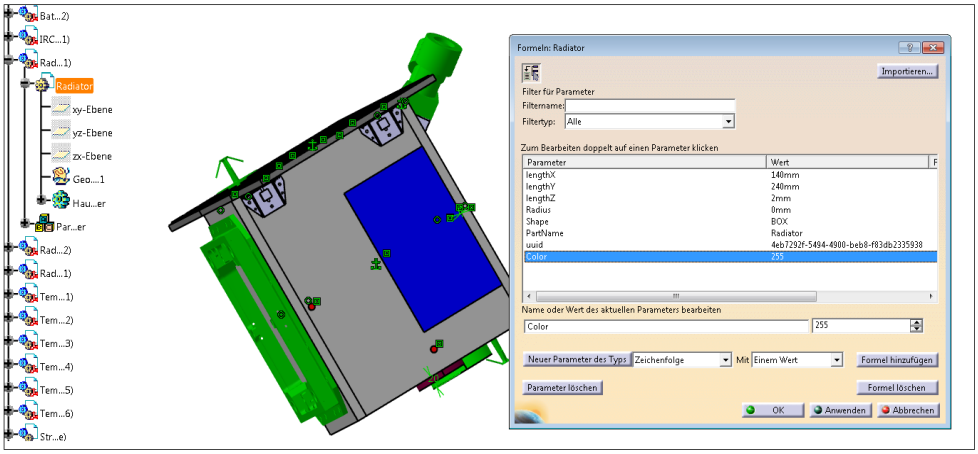

Abbildung 7.11: Änderungen im CAD Modell

Abbildung 7.8 zeigt denselben Radiator nach dem Import der mechanischen Designinformationen. Durch das CAD Modell hat sich das Bauteil in Form und Farbe geändert. Die geänderten Informationen über den Radiator sind auch im json-Format wieder zu erkennen.

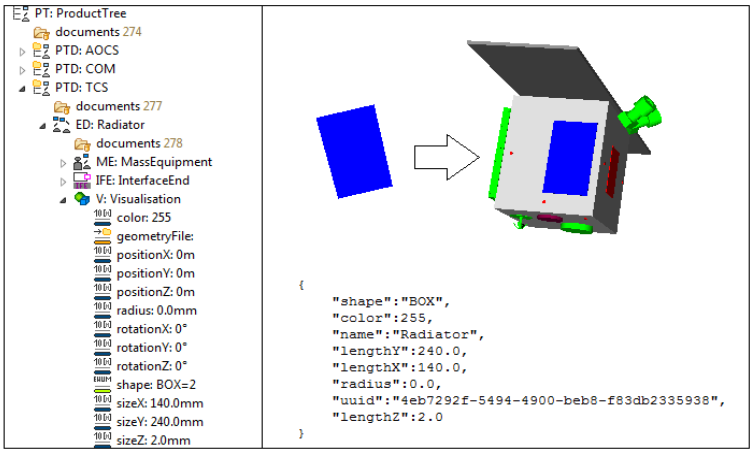

Abbildung 7.12: Radiator nach Import

Abbildung 7.13 zeigt den geänderten Radiator in den unterschiedlichen Konfigurationen in Virtual Satellite nach dem Import auf den Assembly Tree *SatWithIRCam*. Die einzelnen Konfigurationen weisen nun deutliche Änderungen auf und sind nicht mehr identisch. Zu sehen ist, dass durch den Import auf den Assembly Tree das Aussehen des Radiators, sowie dessen Position in der Konfiguration geändert wurde. Durch die Änderung der äußeren Darstellung des Radiators wird nicht nur der importierte Assembly Tree geändert, sondern auch der zweite Assembly Tree und der Configuration Tree. Die Informationen über die

äußere Erscheinung eines Bauteils werden immer im Product Tree hinterlegt. Da von diesem alle erben, besitzen nun alle Trees, den durch das mechanische Design geänderten Radiator. Des Weiteren wurde der Radiator in der Assembly Tree Konfiguration verschoben. Die Änderung über die neue Position des Radiators ist allerdings nur im importierten Assembly Tree *SatWithIRCam* zu erkennen.

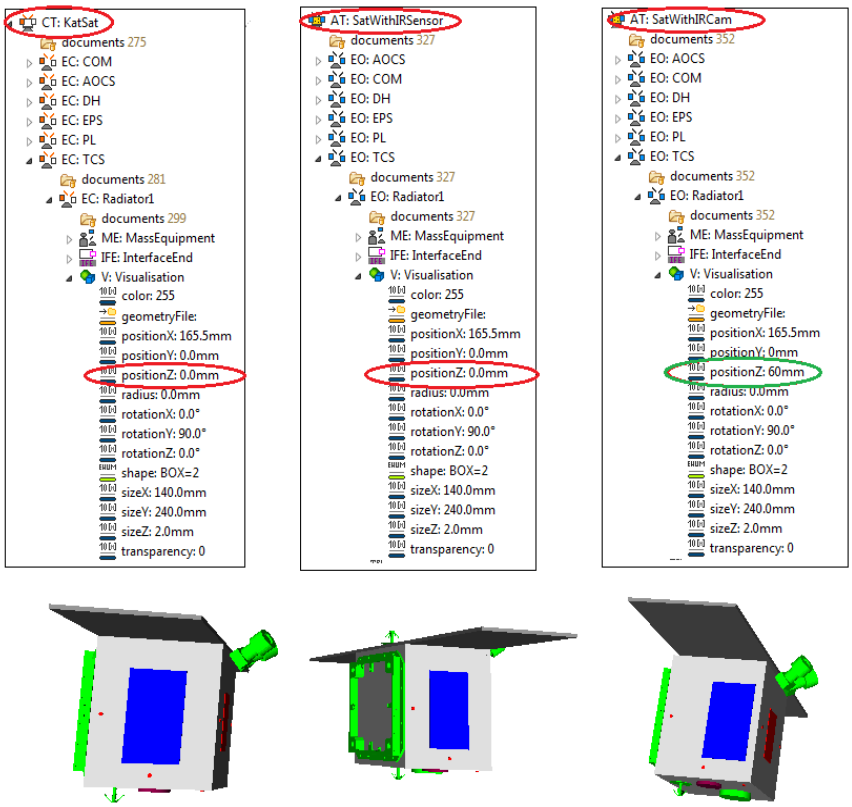

Abbildung 7.13: Trees nach Import

Die Änderungen eines Bauteils im Assembly Tree haben Auswirkungen auf alle Konfigurationen des Satelliten mit der Begründung, dass der Assembly Tree die Konfiguration des tatsächlichen Satelliten zeigt, wie er später auch im Orbit fliegen soll. Sobald sich an diesem Bauteile ändern, sollte das die Bauteile in allen anderen Konfigurationen mit ändern. Um eine Simulatorkonfiguration oder einen zweiten Satelliten zu erstellen dessen Konfiguration etwas anders aufgebaut ist, als die in der ersten Konfiguration des Configuration Tree, ist eine Änderung allein im Assembly Tree ausreichend. Der Configuration Tree ist eine erste Annahme zur Konfiguration des Satelliten und gibt an wie der Satellit aussehen *soll*, während der Assembly Tree zeigt wie der Satellit *tatsächlich* aussieht.

Zusätzliche mechanische Informationen

In CATIA besteht die Möglichkeit das CAD Modell des Satelliten mechanisch sehr fein darzustellen. Beispielsweise können Bohrungen, Gelenke oder Schrauben in das Modell integriert werden. Für den Testdatensatz wurden in CATIA zusätzliche mechanische Bauteile im Modell integriert. Im oberen Bereich zeigt das Schaubild 7.14 eine Bohrung im CAD Modell für die Infrarot Kamera, die Nutzlast des Satelliten, sowie eine Gelenkverbindung und eine Halterungen der beiden Sternensensoren. Die Bohrung und die mechanischen Bauteile sind gelb hervorgehoben. Im unteren Bereich des Schaubilds ist die Virtual Satellite Visualisierung des Satelliten dargestellt. In dieser sind weder Gelenk, Halterung noch Bohrung vorhanden. Gekennzeichnet sind die fehlenden Bauteile durch eine rote Umkreisung. Beim Import des mechanischen Modells in Virtual Satellite werden diese Informationen nicht in der json-Datei berücksichtigt, da sie im CAD Modell angelegt worden sind und somit keine Virtual Satellite UUID besitzen. Nach den CRUD Kriterien ist CATIA nicht in der Lage neue Bauteile zu definieren. Bei einem erneuten Austausch zwischen Virtual Satellite und CATIA bleiben das Gelenk und die Bohrung jedoch im CAD Modell enthalten, da diese Informationen mit im CAD Modell hinterlegt werden. Dasselbe gilt für mechanische Bedingungen, welche im CAD Modell angelegt werden. Diese dienen dem mechanischen Ingenieur um die einzelnen Bauteile miteinander in Verbindung zu bringen.

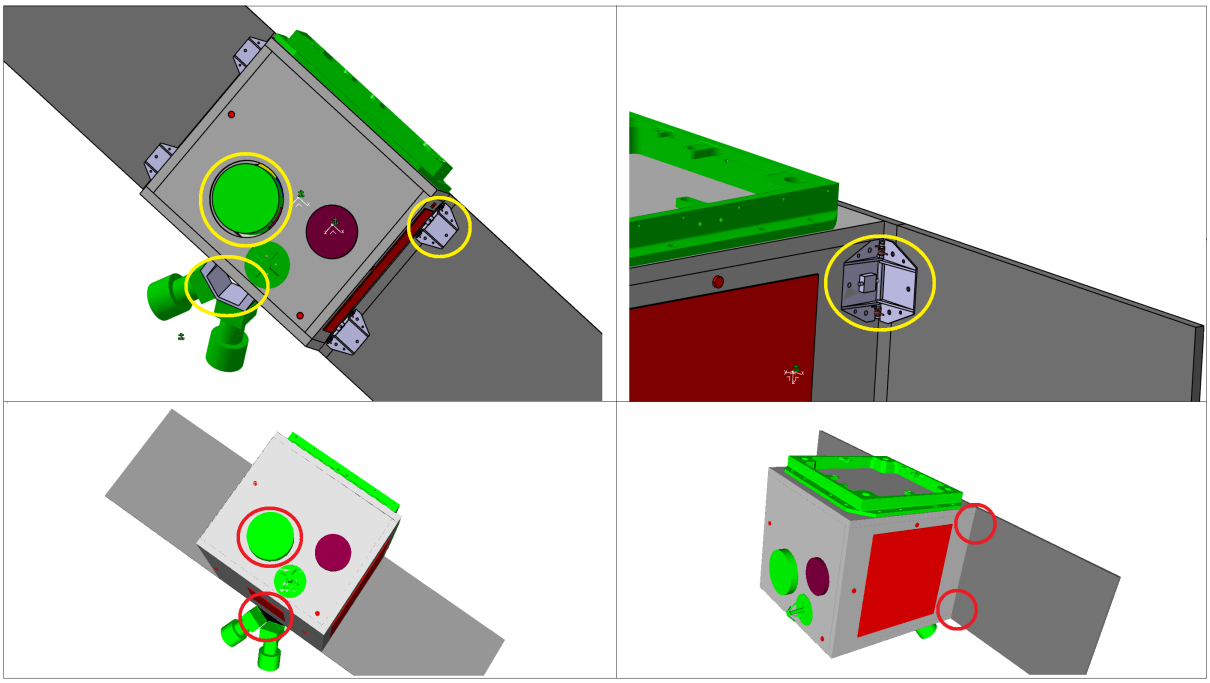

Abbildung 7.14: Bohrung und Gelenk im CAD Modell vs. System Modell

Beim Import der erstellten json-Datei aus CATIA werden Informationen zu strukturellen Parametern, wie Schwerpunkt und Trägheitsmomente mit in das System Modell übergeben. Diese Informationen können mit dem Konzept *Catia Properties* an die vorhergesehenen Konfigurationen und Bauteile des

Satelliten angehängt werden. Abbildung 7.15 zeigt im rechten Bereich den Ausschnitt einer json-Datei mit Informationen zum Massenschwerpunkt eines Configuration Trees. Im linken Bereich sind diese Informationen mit Hilfe des Catia Properties Konzeptes an den Configuration Tree angefügt.

Abbildung 7.15: CATIA Eigenschaften im System Modell

7.3 Ergebnisse

Die vorgestellte Entwicklung des Satelliten in CATIA und in Virtual Satellite in Bezug auf das Roundtrip Engineering stellt einige wichtige Ergebnisse für den Informationsaustausch von mechanischem Design und System Modell dar. Zum Einen erscheinen mechanische Konstruktionsbauteile, wie Gelenke und Schrauben im CAD Modell, jedoch nicht im System Modell von Virtual Satellite. Diese Informationen sind lediglich für den mechanischen Ingenieur von Bedeutung und haben im System Modell keinerlei Bedeutung. Einzig und allein die durch Schrauben oder Bohrungen neu erforderte Position der Komponente in der Konfiguration spielt für das System Modell eine Rolle. Dasselbe gilt für die Gelenkverbindungen der Solarpanele am Strukturgerüst und mechanische Bedingungen, auch diese sind nur im CAD Modell vorhanden. (vgl. Abbildung 7.14) In diesem Fall handelt es sich um einfache Gelenkverbindungen zum Ausklappen der Solarpanele welche nur einmalig genutzt werden. Im Falle von komplexeren Gelenken oder sogar Stepper Motoren, die ein automatisiertes Ausklappen gewährleisten, müsste darüber nachgedacht werden, diese Information mit ins System Modell zu übernehmen, da ein solches Bauteil Auswirkungen auf das Power und das Thermalsystem haben könnte.

Während das System Modell in Virtual Satellite sich im Laufe des Entwurfsprozesses weiter verfeinert durch State Machines, Power Equipment, Massen Equipment und Interface Verbindungen, verfeinert das CATIA Modell das konstruktive mechanische Design des Satelliten, durch mechanische Bedingungen, Gelenkverbindungen, Bohrungen etc. Beide Modelle können im Entwurfsprozess getrennt voneinander

verfeinert werden. Durch die bidirektionale Schnittstelle können die Modelle jedoch Veränderungen aus dem jeweilig anderen Modell bei Relevanz übernehmen oder selbst Änderungen am anderen Modell vornehmen. Somit entsteht ein Round Trip Engineering von dem nicht nur beide Modelle profitieren, sondern auch die anderen Domänen des Raumfahrzeugsystems.

Das Roundtrip Engineering in Bezug auf einen Assembly Tree hat den Vorteil, dass in Virtual Satellite der Assembly Tree als tatsächliche Konfiguration des Satelliten für Simulatorkonfigurationen genutzt werden kann. Durch den Austausch mit dem mechanischen Design entsteht auch für den Assembly Tree ein sehr komplexes und detailliertes Modell des Satelliten, welches an die Simulatorkonfiguration übergeben werden kann. Dabei gilt, je detaillierter das Simulationsmodell, desto präziser die Ergebnisse der Simulation.

Kapitel 8

Zusammenfassung

In dieser Arbeit wurde anhand der CRUD Kriterien der Informationsfluss zwischen mechanischem Design und MBSE umgesetzt. Dabei wurde der Informationsfluss analysiert, um ausschließlich relevante Daten an das jeweilig andere Modell zu übertragen. Da das System Modell das zentralisierte Modell darstellt, auf welches alle Domänen Zugriff haben, ist dieses in der Lage Informationen zu Satellitenbauteilen hinzuzufügen, zu löschen, zu ändern und zu ersetzen. Aus diesem Grund wurden im System Modell in Virtual Satellite alle CRUD Kriterien implementiert. Im mechanische Modell in CATIA jedoch wurden nur die CRUD Kriterien Replace und Update implementiert, um Bauteile geometrisch zu verändern oder deren Lage in der Satellitenkonfiguration zu ändern. Um den Austausch zwischen mechanischem Design und MBSE zu ermöglichen, muss gewährleistet werden, dass der erste Austausch im Roundtrip Engineering dem Erzeugen eines System Modells nachgeht und den erstmaligen Export dessen in das mechanische Design beinhaltet. Erst dann kann das Roundtrip Engineering erfolgen. Anhand eines Vergleichs mehrerer Austauschformate wurde mit dem json-Format ein geeignetes Format gewählt, um den Informationsfluss zu realisieren.

Der Informationsaustausch zwischen MBSE und mechanischen Modellen impliziert diverse Vorteilen, welche für Systeme, insbesondere Raumfahrtsysteme, im kompletten Lebenszyklus genutzt werden können. Das System Modell und auch das mechanische Modell verfeinern sich im Laufe des Entwurfsprozesses. Durch das Roundtrip Engineering, welches durch die hier integrierte bidirektionale Schnittstelle entsteht, können die voneinander getrennten Modelle miteinander verknüpft und die Änderungen des jeweils anderen Modells übernommen werden, ohne dass die jeweiligen Verfeinerungen der beiden Seiten verloren gehen. Das ermöglicht ein detaillierteres und präziseres Systemdesign. Des Weiteren wird mit dem json-Format ein Austauschformat gewählt, welches in modernen Programmiersprachen verwendet wird und nicht nur für den Austausch von CAD Informationen geeignet ist. Dadurch entsteht eine Austauschmethode, welche auch für andere Domänen in Frage kommt. Der Austausch zeigt, dass zum einen durch das komplexere und detailliertere System Modell Simulatoren konfiguriert werden können. Die Simulationsmodelle können direkt aus dem System Modell entnommen werden und müssen nicht im Simulation Tool selbst erstellt werden. Auf der einen Seite werden dadurch Arbeitsprozesse im Entwurf erleichtert und auf der anderen Seite werden so Fehler vermieden, welche aufgrund von manueller Konfiguration der Simulatoren auftauchen können. Zum anderen können andere Domänen über das System Modell auf mechanische Designinformationen zugreifen und diese dort unter Umständen auch ändern. Dadurch entsteht ein flüssiger Austausch im gesamten System, der nicht nur in der Designphase, sondern auch in den späteren Lebenszyklusphasen für Tests und Simulationskonfigurationen besteht.

Kapitel 9

Fazit und Ausblick

Der bidirektionale Austausch zwischen System Modell und mechanischem Design Modell für den modellbasierten Systementwurf in der Raumfahrt bringt einige Vorteile mit sich. Die hier vorgestellte bidirektionale Integration kann als Grundgerüst des bidirektionalen Austauschs zwischen MBSE und mechanischem Design aufgefasst werden.

In dieser Arbeit kann in der MBSE Software Virtual Satellite das System Modell eines Raumfahrzeugs durch die Integration der CAD Software CATIA auf mechanischem Niveau sehr detailliert beschrieben und durch wichtige mechanische Parameter ergänzt werden. CATIA bietet zudem weitere Vorteile. Das CAD Tool ist in der Lage, kleinere Bewegungen im mechanischen Modell darzustellen. Somit kann z.B. der Mechanismus zum Ausklappen der Solarpanele des Satelliten dargestellt werden. Auch das Justieren einer Kamera oder eines Sensors kann sich mit Hilfe von CATIA darstellen lassen. Das führt dazu, dass mit Hilfe der bidirektionalen Schnittstelle, auch definierte Betriebszustände des Raumfahrzeuges visualisiert und dargestellt werden können. Durch die Möglichkeit den Satelliten nicht nur als ein System zu betrachten, sondern als ein System in mehreren Betriebszuständen mit mehreren Konfigurationen würde ein deutlicher Vorteil im Verständnis des Systems zu Stande kommen, der von allen Domäneningenieuren genutzt werden kann.

Des Weiteren besteht die Möglichkeit zusätzlich zu den mechanischen Designparametern in CATIA auch auf thermische Parameter des Materials eines CATParts zugreifen zu können. CATIA besitzt eine *Thermal Analysis Workbench* mit der materialspezifische Wärmeparameter wie, Wärmekapazität, Wärmeleitfähigkeit etc., dem Material angefügt werden können. Auf dieser Basis können in CATIA auch Thermalanalysen durchgeführt werden, um die Temperaturverteilung eines Bauteils oder Assemblies zu bestimmen. Um jedoch, besonders mit Betrachtung auf Raumfahrzeuge, eine genaue Thermalanalyse zu erhalten sind Tools wie *ESATAN* besser geeignet. Diese beziehen zusätzlich zum Raumfahrzeug ebenfalls den Satellitenorbit, sowie Schatten- und Sonnenphasen in die Thermalanalyse ein. Dennoch benötigt ESATAN ein geeignetes Modell des Raumfahrzeuges für die Analyse. Durch die Integration des mechanischen Designs an das System Modell ist es nun möglich, dass die Thermalsystem Domäne aus dem System Modell heraus die wichtigen Informationen aus dem mechanischen Modell übernehmen, um damit später die Simulation durchzuführen kann. Ansonsten müsste in ESATAN ein komplett neues Modell erstellt werden. Um diese doppelte Arbeit zu umgehen, wäre es ebenfalls denkbar eine Schnittstelle vom System Modell aus an ein Thermalanalyse Tool wie ESATAN zu erstellen, welche die mechanischen Design Informationen, die sich im System Modell befinden, direkt an ESATAN weitergeben kann. Es besteht des Weiteren die Möglichkeit der Einbindung der *Electrical Wire Harness Workbench* und der *Structural Analysis Workbench*. Virtual

Satellite ist bereits in der Lage Verkabelungen und Schnittstellen der Bauteile zu definieren, kann diese jedoch nicht visualisieren. Durch die Verbindung zur *Electrical Wire Harness Workbench* ist die Visualisierung im CAD Modell gegeben. Damit kann festgelegt werden welche Bauteile miteinander verbunden werden und wo die Verbindungen und Kabel in der Satellitenkonfiguration lang führen.

Die bidirektionale Kopplung von Virtual Satellite und CATIA sollte weiter untersucht werden, wenn es darum geht Änderungen zu importieren, die zu Konflikten führen können. Wurde sowohl im CAD als auch im System Modell, dasselbe Bauteil geändert, sollte der zuständige Domäneningenieur selbst entscheiden dürfen, welche Änderungen er beibehalten möchte. Der jetzige Stand der Implementierung der bidirektionalen Kopplung befasst sich nicht mit dieser Konfliktstellung und überschreibt durch den Import eines Modells die anderen Modell Änderungen. Des Weiteren muss im Nutzerrechtmanagment des Virtual Satellite geklärt werden, welche Änderungen das CAD Modell am System Modell vornehmen kann. Der derzeitige Stand sieht es vor, dass jeder Domänenexperte nur auf seinem Gebiet Informationen ändern, hinzufügen bzw. löschen kann. Daher stellt die entworfene bidirektionale Integration von mechanischem Design an den modellbasierten Systementwurf nur ein Grundgerüst dar, welches leicht erweiterbar ist. Des Weiteren kann der Entwurf der bidirektionalen Schnittstelle für andere Domänen als eine erste Idee dienen, um z.B. eine direkte Schnittstelle zwischen dem Thermalanalyse Tool ESATAN und dem System Modell zu verwirklichen.

Literaturverzeichnis

[1] em engineering methods AG. *Model Based Systems Engineering: Prinzipen, Anwendung, Beispiele, Erfahrung und Nutzen aus Praxissicht*. In: *Whitepaper 08/16*. 2016.

[2] R. Blien. *CAD Glossar*. 21. März 2018. URL: `www.blien.de/ralf/cad/db`.

[3] M. Campbell-Kelly. „An Introduction to Macros“. In: *Macdonald & Co* (1973).

[4] Materialise Cloud. *STL Converter*. 13. März 2018. URL: `https://cloud.materialise.com/tools/stl-converter`.

[5] Dr. D. Cowper. „Systems Engineering and Project Management Integration: V-model (Explanatory Note)“. In: *APM/INCOSE Joint Working Group* (2014).

[6] D. D'Ascanio. „Analysis of the ARIEL spacecraft thermal architecture in the early A Phase“. Master Thesis. School of Industrial, Information Engineering, Department of Aerospace Science und Technology, 2016.

[7] DLR. *FireBIRD satellites – a duo on course for early fire detection*. Hrsg. von P. Burtscheidt und S. Kaufhold. 8. Feb. 2018. URL: `http://www.dlr.de`.

[8] eclipse. *The Eclipse Foundation open source community*. 19. März 2018. URL: `http://www.eclipse.org/`.

[9] European Space Agency ECSS. *Space Engineering - Space system data repository (Technical Memorandum ECSS-E-TM-10-23A)*. Techn. Ber. 2011.

[10] S. Erler. „Aufbereitung von Produktmodellen der Industrie zur Verwendung in interaktiven 3D - Anwendungen“. Diplomarbeit. Technische Universität Dresden, 2012.

[11] esa. *European Ground Systems Common Core - EGS-CC Concepts*. 16. März 2018. URL: `http://www.egscc.esa.int/concepts.html`.

[12] P. Fischer. *Virtual Satellite - Zukunft des Model-Based Systems Engineering in der Raumfahrt*. 2017. URL: `http://www.dlr.de/sc/desktopdefault.aspx/tabid-5135/8645_read-8374/`.

[13] P. M. Fischer u. a. „Conceptual Data Model - A Foundation for Succesful Concurrent Engineering“. In: *CERA - Concurrent Engineering : Research and Application* (2017).

[14] P. M. Fischer u. a. „Implementing model-based system engineering for the whole lifecycle of a spacecraft“. In: *CEAS Space*. 2017.

[15] J. Fuchs. *The "Virtual Spacecraft Design"*. Hrsg. von esa. 20. Okt. 2017. URL: `http://www.vsd-project.org/`.

[16] S. Ghanta. „A STEP Implementation For Product Data Exchange“. Master Thesis. Mid Sweden University - The Department of Information Technology und Media (ITM, 2012.

[17] C. F. Goldfarb und P. Prescod. *XML Handbuch*. 1999.

[18] P. Graignic u. a. „Complex System Simulation - Proposition of a MBSE Framework for Design-Analysis Integration“. In: *CSER*. 2013.

[19] Dipl.Ing. J. Groß. „Aufbau und Einsatz von Entwurfssprachen zur Auslegung von Satelliten“. Diss. Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen Universität Stuttgart, 2014.

[20] L. E. Hart. „Introduction To Model-Based System Engineering (MBSE) and SysML“. In: *Delaware Valley INCOSE Chapter Meeting*. 30. Juli 2015.

[21] INCOSE. *What is Systems Engineering?* 25. März 2018. URL: `https://www.incose.org/`.

[22] OMG INCOSE. *OMG Systems Modeling Language (OMG SysML™)Tutorial*. Hrsg. von Sanford Friedenthal, Alan Moore und Rick Steiner. 2009.

[23] ecma International. „The JSON Data Interchange Syntax“. In: *ecma International* (2017).

[24] C. Iwata u. a. „Model-Based Systems Engineering in Concurrent Engineering Centers“. In: *SPACE Conferences and Exposition*. 2015.

[25] D. Steffen J. Gausemeier R. Dumitrescu. *Systems Engineering in der industriellen Praxis*. 2013.

[26] S. Rudolph J. Groß. „Hierarchie von Entwurfsentscheidungen im modellbasierten Entwurf komplexer Systeme“. In: *Tag des Systems Engineering*. 2011.

[27] S. Rudolph J.Gross. „Geometry and Simulation Modeling in Design Languages“. In: (2016).

[28] R. Johnson und B. Woolf. „The Type Object Pattern“. In: *Type Object* (1997).

[29] C. Lange u. a. „Systematic reuse and platforming: Application examples for enhancing reuse with model-based systems engineering methods in space systems development“. In: *Concurrent Engineering: Research and Applications* (2017).

[30] P. Leach, M. Mealling und R. Salz. *A Universally Unique IDentifier (UUID) URN Namespace*. Software. Network Working Group, 2005.

[31] Y. Liu u. a. „Realtime Immersive Visualization for Satellite Configuration and Version Comparison“. In: *SESP 2017*. 2017.

[32] B. Cole M. Bajaj D. Zwemer. „Architecture to Geometry – Integrating System Models with Mechanical Design“. In: *American Institute of Aeronautics and Astronautics* (2016).

[33] J. Mueller. „Bidirektionale Anbindung eines grafischen Editors an ein Systemmodell“. Bachelor Thesis. Duale Hochschule Baden-Württemberg, 2013.

[34] A. Gerndt P. M. Fischer V. Schaus. „Design Model Data Exchange between Concurrent Engineering Facilities by Means of Model Transformation“. In: *13th NASA-ESA Workshop on Product Data Exchange* (2011).

[35] Andreas Gerndt P. M. Fischer R. Wolff. „Collaborative Satellite Configuration Supported by Interactive Visualization“. In: *IEEEAC Paper* (2012).

[36] Joachim Fuchs P. M. Fischer Harald Eisenmann. „Functional Verification by Simulation based on Preliminary System Design Data“. In: *6 th International Conference on Systems & Concurrent Engineering for Space Applications*. 2014.

[37] M. Pecchioli und J. M. Carranza. „The Main Concepts of the European Ground Systems – Common Core (EGS-CC)“. In: (2013).

[38] M. J-L. C. Poucet. „Phase-B Thermal Control Subsytem Design for the ESEO Satellite“. Politecnico di Milano, Facoltà di Ingegneria Industriale, 2012.

[39] J. Rey. *Modeling with VSEE: Definition of Guidelines and Exploitation of the Models*. Modeling with VSEE: Definition ofGuidelines and Exploitation of the Models. European Space Agency, 2013.

[40] D. L. Rosenband. *A Remote Procedure Call Library*. 2. Aug. 2018. URL: `http://web.mit.edu/6.033/1997/reports/dp1-danlief.html`.

[41] V. D'Souza S. Zunke. „JSON vs XML: A Comparative Performance Analysis of Data Exchange Formats“. In: *Volume 3, Issue 4, August 2014* (2014).

[42] V. Schaus u. a. „Collaborative Satellite Configuration Supporting CATIA Export“. In: *SECESA*. 2012.

[43] A. Schledermann. *STL-Dateiformat*. 2017.

[44] H. Schumann, P. Zamov und S. Escher. „Model-based design and tool data exchange in aerospace: a case study“. In: *German Aerospace Congress, Bremen* (2011).

[45] G. G. Slabaugh. *Computing Euler angles from a rotation matrix*. 2018.

[46] SoftSelect. *Definition Plug-In*. 25. März 2018. URL: `http://www.softselect.de/business-software-glossar/plug-in`.

[47] H. Stachowiak. *Allgemeine Modelltheorie*. 1973.

[48] Dassault Systemes. *Dassault Systemes Homepage*. 22. März 2018. URL: `https://www.3ds.com/de/`.

[49] ESATAN TMS. *ESATAN Thermal modelling suite*. 9. März 2018. URL: `https://www.esatan-tms.com/products/catdescription.php?ID=4`.

[50] C.-O. Truica, A. Boicea und I. Trifan. „CRUD Operations in MongoDB“. In: *International Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013)* (2013).

[51] A. Weiß, V. Maiwald und G. Wübbels. „Concurrent Evaluation - an Application for DLR's Concurrent Engineering Facility“. In: *SECESA* (2010).

[52] J. R. Wertz. *Space Mission Engineering: The new SMAD*. 2011.

[53] F. Westphal. *Unit Tests mit JUnit*. Hrsg. von Frank Westphal. 19. März 2018. URL: `http://www.frankwestphal.de`.